Internet Programming

SmartPhone Apps

Dan Brandon, Ph.D., PMP

Mobile Phones

Mobile phones are now the most common device used for browsing the web!

Mobile Application Development

- Lots of jobs - 34% growth rate for next 10 years
- Starting salaries of $90,000 thru $130,000

User Experience: Desktop vs Mobile

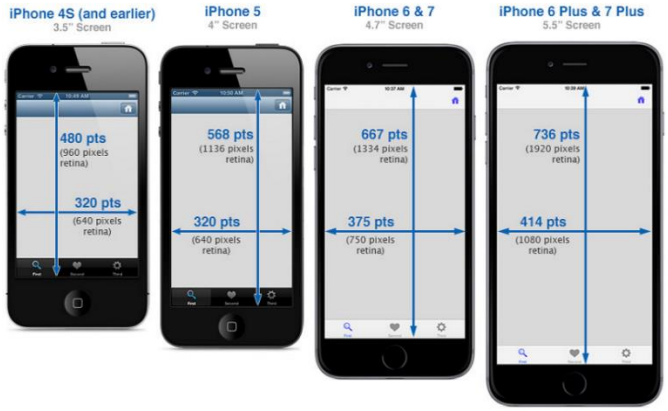
User Experience	Mobile	Desktop
Page Content	Content should be short and to the point.	Content can be extensive, giving readers the opportunity to explore all facets of the topic.
Page Layout	Content should be laid out within a single column with no horizontal scrolling.	With a wider screen size, content can be more easily laid out in multiple columns.
Hypertext Links	Links need to be easily accessed via a touch interface.	Links can be activated more precisely using a cursor or mouse pointer.
Network Bandwidth	Sites tend to take longer to load over cellular networks and thus overall file size should be kept small.	Sites are quickly accessed over high-speed networks, which can more easily handle large file sizes.
Lighting	Pages need to be easily visible in outdoor lighting through the use of contrasting colors.	Pages are typically viewed in an office setting, allowing a broader color palette.
Device Tools	Mobile sites often need access to devices such as phone dialing, messaging, mapping, and built-in cameras and video.	Sites rarely have need to access desktop devices.

Smartphone Key Features

- Portability
- Phone/SMS (text messaging)
- Screen Size
- Touch Screen
- Look & Feel
- Button Size & Behavior
- Icon Size and Behavior
- Portrait and Landscape Modes
- Multimedia: camera, microphone, video-cam
- GPS & Geolocation
- Sensors: three-axis gyro, accelerometer, proximity sensor, ambient light sensor

Screen Size

- The technology discussed here for smartphones also applies to other portable smart devices such as:
  - iPads at 9.7in
  - Samsung's Galaxy Note
  - Amazon's Kindle
  - Dell Streak
  - iPad Mini
- iPhone 4 screen is 3.6 in (diagonal)
- iPhone 5 screen is 4 in (diagonal)
- iPhone 6 screen is 4.7 in (diagonal)
- iPhone 6+,7 screen is 5.5 in (diagonal)
- iPhone X is 5.8 in (diagonal)



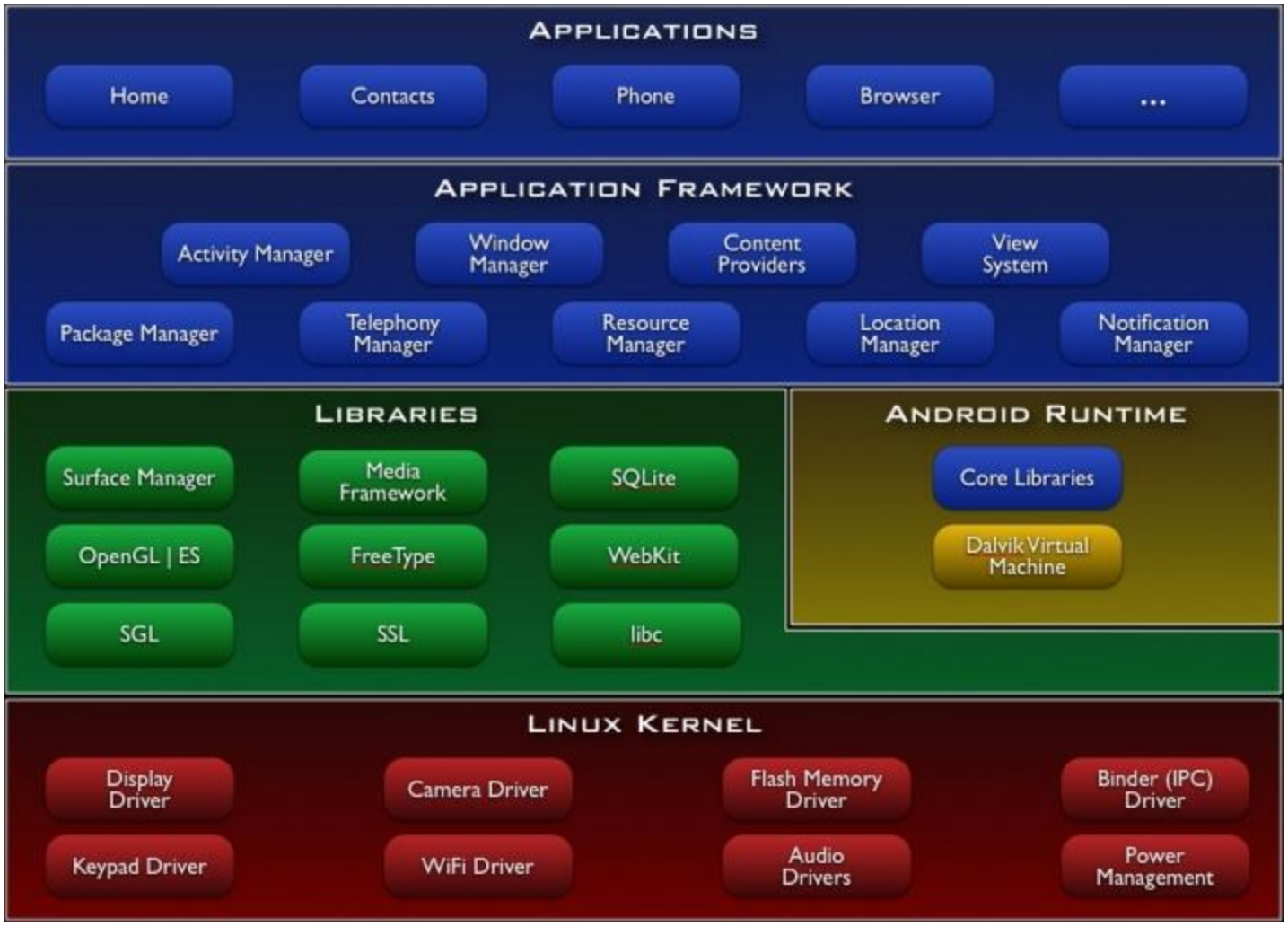
SmartPhone Development Choices

- Vendor Proprietary API/SDK
  - Apple (iOS)
  - Android
  - Blackberry
  - Others

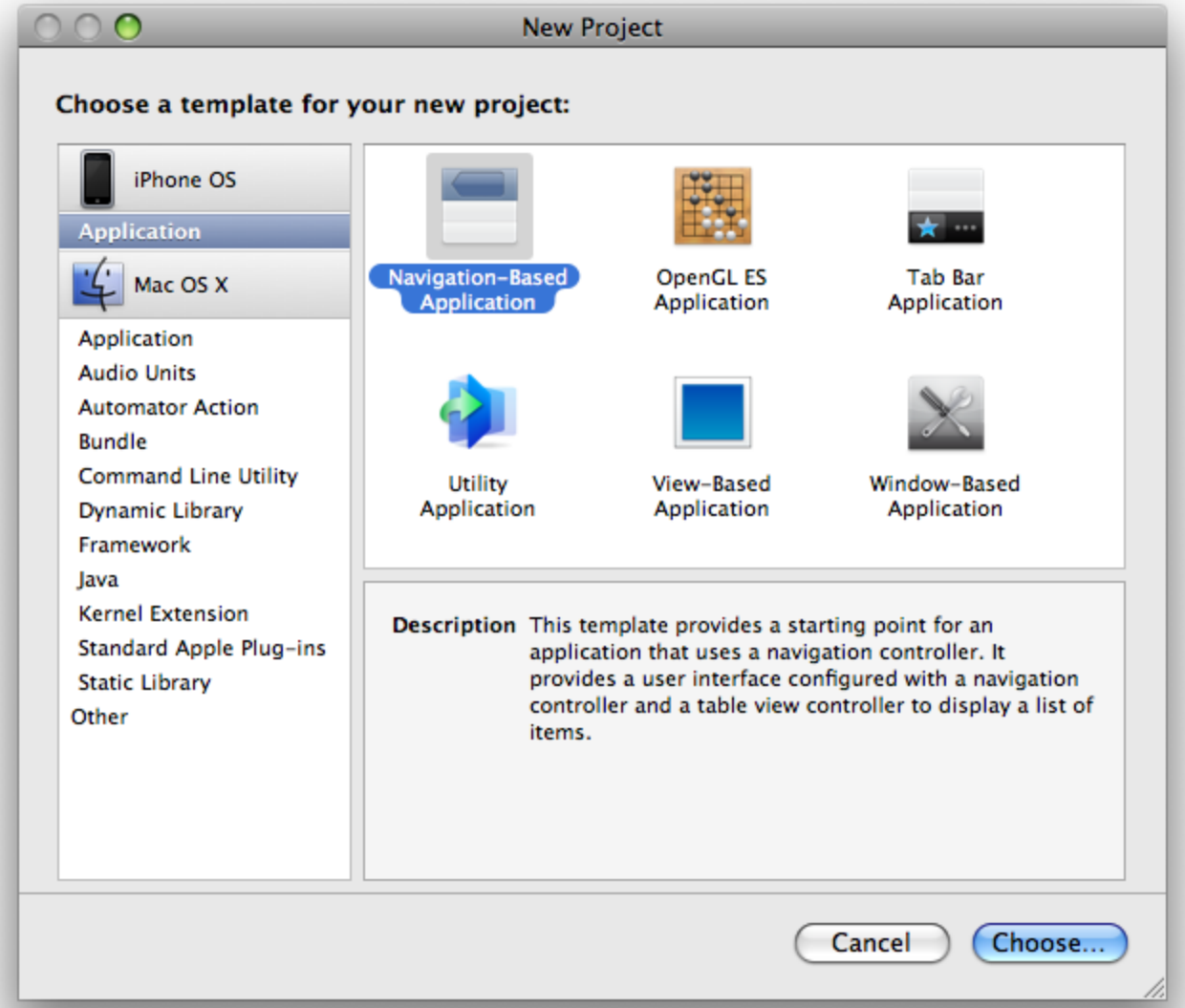
Android SDK	Software development kit for Android developers(*developer.android.com/sdk*)
iOS SDK	Software development kit for iPhone, iPad, and other iOS devices(*developer.apple.com*)
Opera Mobile SDK	Developer tools for the Opera Mobile browser(*www.opera.com/developer*)

- Open Source HTML5, JavaScript, CSS
  - Manually
  - Using a UI foundation (i.e. jQuery Mobile)
  - WYSIWYG (i.e. Yapp, VIX, etc.)
- Build prototype with HTML5 et. al. , then convert to native API via PhoneGap

Android API/SDK
[Java plus C/C++ libraries]



iOS Platform/SDK [Objective-C]



iOS Swift

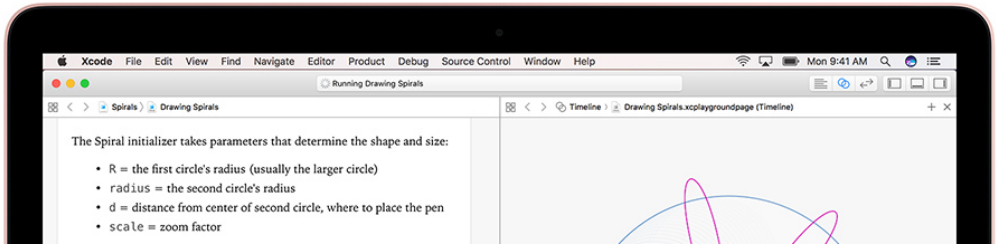
Developer Discover Design Develop Distribute Support Account

Swift Swift Playgrounds Resources Blog

Swift 3

The powerful programming language that is also easy to learn.

Swift is a powerful and intuitive programming language for macOS, iOS, watchOS and tvOS. Writing Swift code is interactive and fun, the syntax is concise yet expressive, and Swift includes modern features developers love. Swift code is safe by design, yet also produces software that runs lightning-fast.



PhoneGap


PhoneGap
About
Developers
Community
Support
Download
PhoneGap 1.4.1
THE ONLY OPEN SOURCE MOBILE FRAMEWORK THAT SUPPORTS 7 PLATFORMS
Single Code Base
PhoneGap
Deploy to Multiple Platforms
PhoneGap is an HTML5 app platform that allows you to author native applications with web technologies and get access to APIs and app stores. PhoneGap leverages web technologies developers already know best... HTML and JavaScript. Learn More »
PhoneGap 1.4
Nitobi Acquired
PhoneGap Book
PhoneGap Build
PhoneGap + DW
PhoneGap Shop
palm
js
css
html
pricing announced!
PhoneGap Build
Write Once
Compile in the Cloud
Run Anywhere
Learn More
BLOGS
TWITTER
NEWSLETTER
Case Studies
Blog
FAQs
Contact

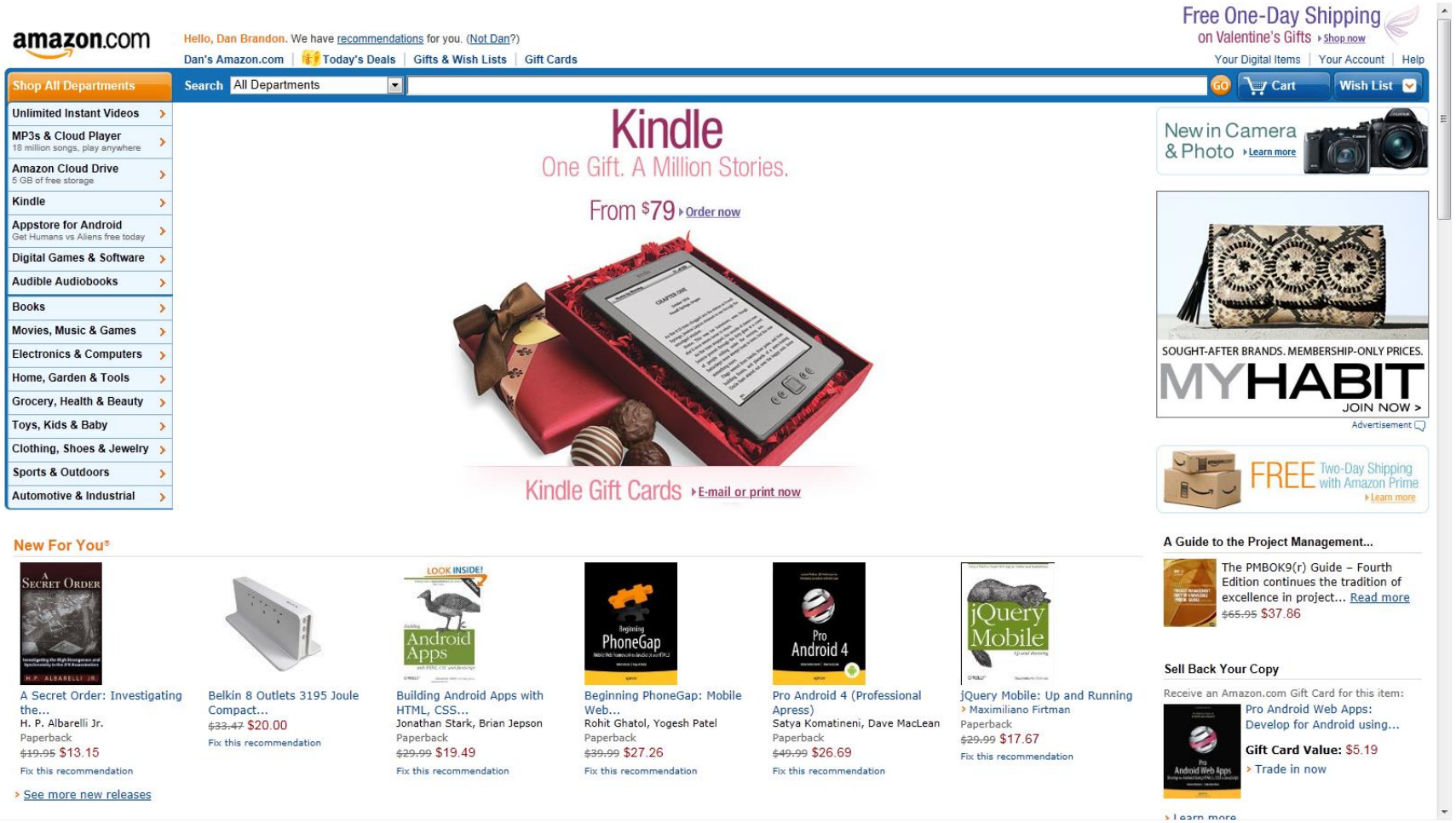
Advantages of HTML5 Based SmartPhone Apps

- Open Source
- Runs on all (most all) platforms
- Don’t have to learn native language, API, and IDE
- Don’t have to contend with Apple’s App Store (iTunes) to get your work released
- Much easier software support and maintenance

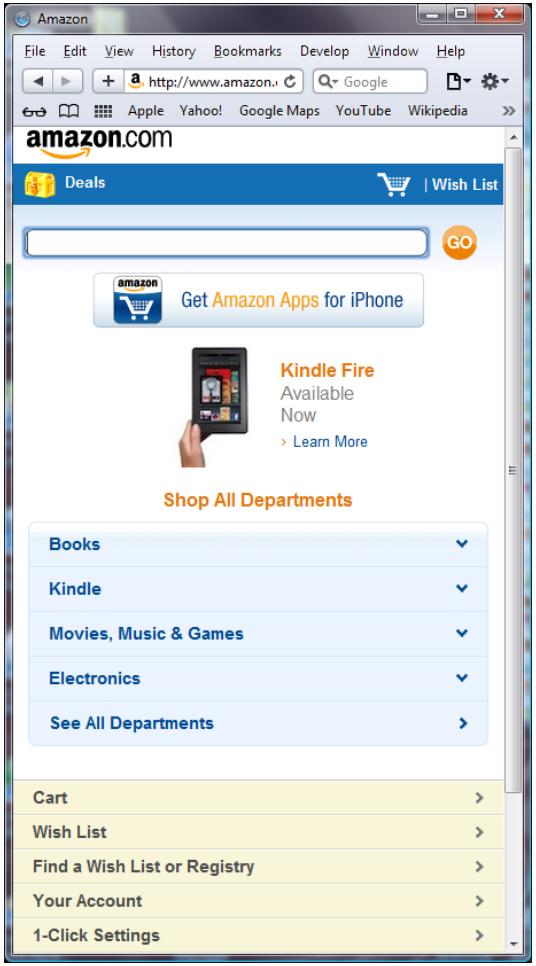
Responsive Design Theory

- The three primary components of responsive design theory identified by Ethan Marcotte are:
  - **flexible layout** so that the page layout automatically adjusts to screens of different widths
  - **responsive images** that rescale based on the size of the viewing device
  - **media queries** that determine the properties of the device rendering the page so that appropriate designs can be delivered to specific devices

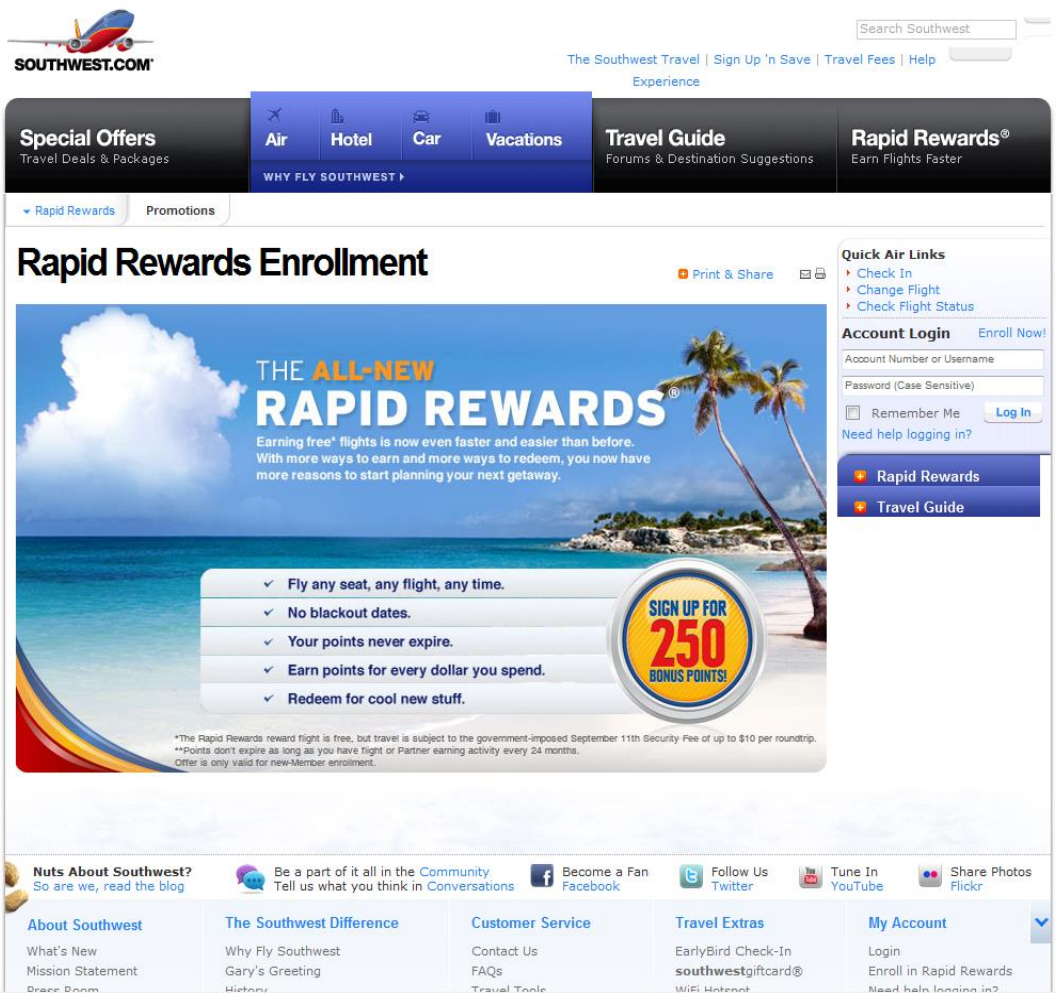
Amazon.com in PC Browser



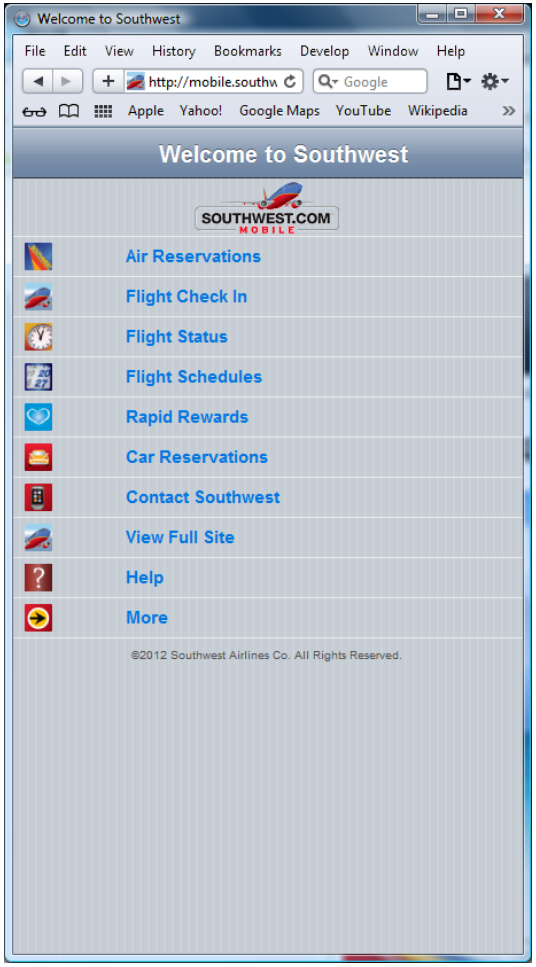
Amazon.com on SmartPhone



SW Air in PC Browser



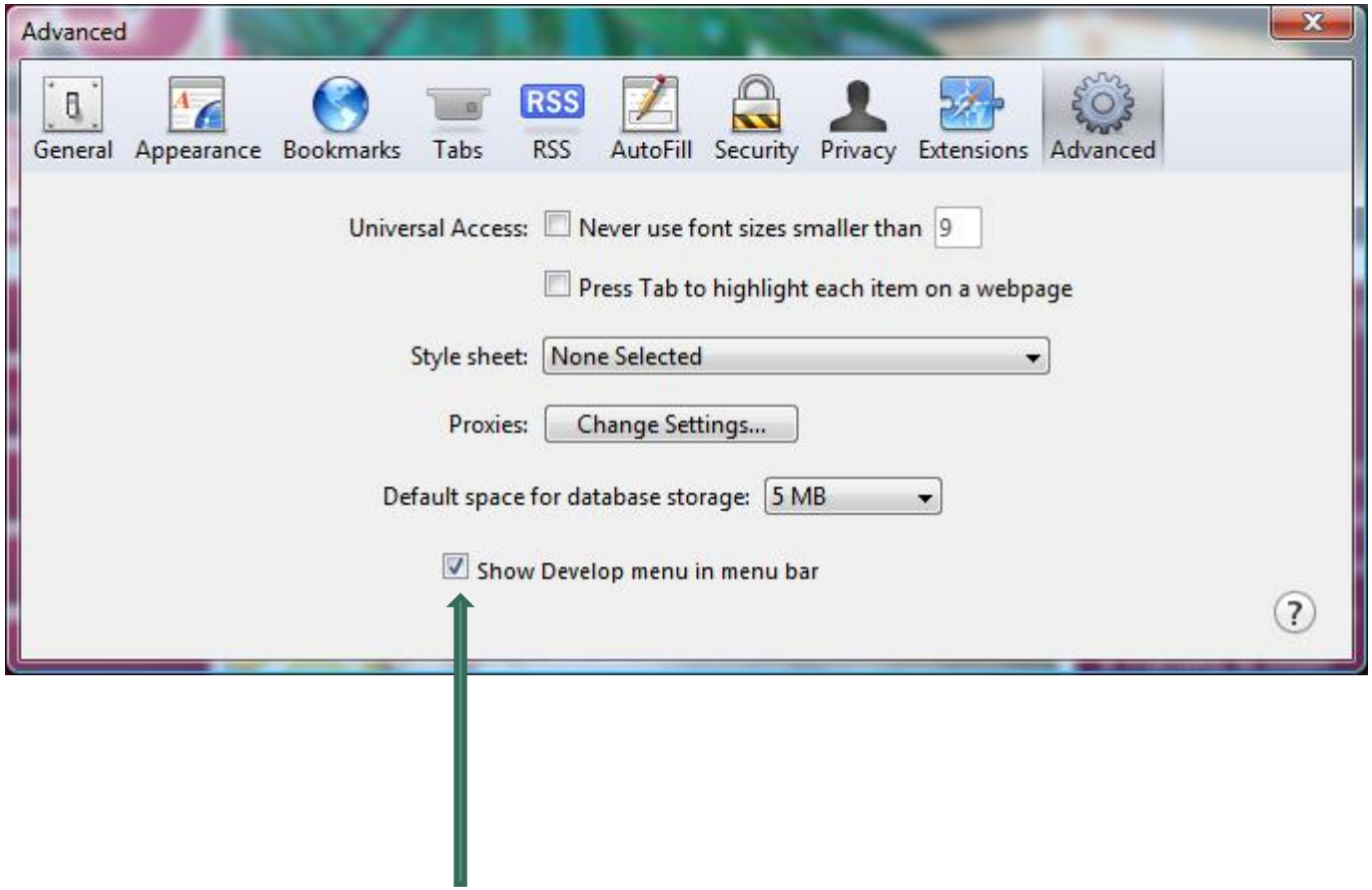
SW Air on SmartPhone



Testing Mobile Apps

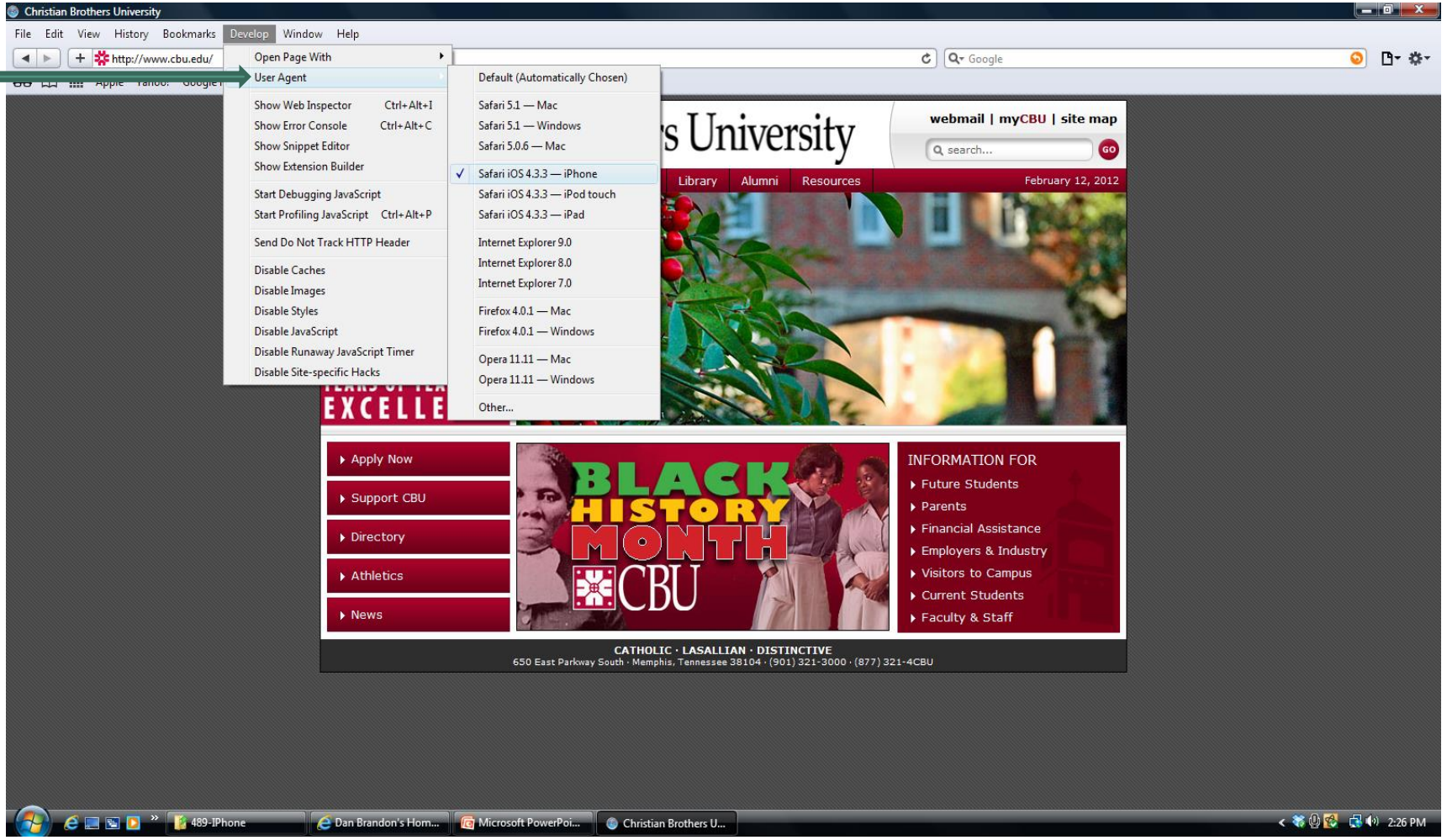
- Using specific smartphones and/or tablets
- Safari browser set for iPhone user agent
- Smartphone simulators/emulators
- Remote labs
  - http://www.deviceanywhere.com
  - http://www.perfectmobile.com

Setting User Agent in Safari
[select advanced preferences]



Setting User Agent in Safari (con't)

[Need to restore down button after page loads]



http://www.mobilexweb.com/emulators

Mobile Emulators & Simulators: The Ultimate Guide

Like 463 people like this. Sign Up to see what your friends like. +1 91 Tweet 1,478

The most useful tools for mobile web are emulators and simulators. In Chapter 4 of Programming the Mobile Web I've made an extensive description and installation guideline for creating a desktop mobile testing environment. And in Chapter 13, I've also reviewed other solutions for mobile testing, including four device remote lab solutions.

I've mixed all this information to create this ultimate guide to **37 download resources** for hundreds of emulators and simulators. Enjoy!

*This list includes content from **Programming the Mobile Web book** with frequent updated information & links.*

LIST OF MOBILE AND TABLET EMULATORS FOR MOBILE WEB DESIGN & DEVELOPMENT TESTING

NAME	OFFICIAL	PLATFORM	TYPE	BROWSER TESTING	NATIVE TESTING	COMPATIBILITY
IOS SIMULATOR	OFFICIAL	IOS	SIMULATOR	SAFARI ONLY	OBJECTIVE-C	
DOWNLOAD 3.7GB (LOGIN REQUIRED)		***DEVICES**: IPHONE 3GS, IPOD TOUCH, IPHONE 4, IPAD (TABLET)*				
		COMES WITH XCODE AND NATIVE SDK. YOU CAN'T EMULATE ACCELEROMETER/GYROSCOPE **(DEVICEMOTION API)**. YOU CAN'T EMULATE URI-SCHEMES, SUCH AS **CLICK-TO-CALL**. AS A SIMULATOR, IT DOESN'T PROVIDE AN APPSTORE; YOU CAN'T INSTALL OTHER BROWSERS FOR TESTING, SUCH AS OPERA MINI OR SKYFIRE.				
ANDROID EMULATOR	OFFICIAL	ANDROID	EMULATOR	ANDROID BROWSER – OTHERS	JAVA	
DOWNLOAD 20MB AND 60MB PER PLATFORM PACKAGE		***DEVICES**: GENERIC DEVICES USING 1.1, 1.5, 1.6, 2.0, 2.1, 2.2, 2.3, 3.0 O.S. PLATFORM*				
		WE NEED TO DOWNLOAD IMAGES OF THE PLATFORMS AFTER DOWNLOADING THE SDK. LOOK AT **CHAPTER 4 OF THE BOOK** FOR DETAILS. AFTER DOWNLOADING THE PLATFORM, YOU CAN INSTALL **FIREFOX**, **OPERA MINI**, **OPERA MOBILE**, **SKYFIRE** AND **UCWEB** IN YOUR ANDROID EMULATOR FOR TESTING. YOU CAN DOWNLOAD MOTOROLA, SAMSUNG AND NOOK				

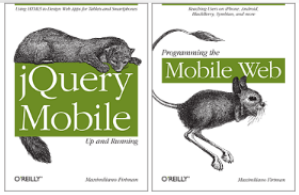
Thoughts, researchs and samples about mobile web by Max Firtman, speaker, trainer and author of O'Reilly books Programming the Mobile Web and jQuery Mobile.

Follow me @firt

Follow this blog

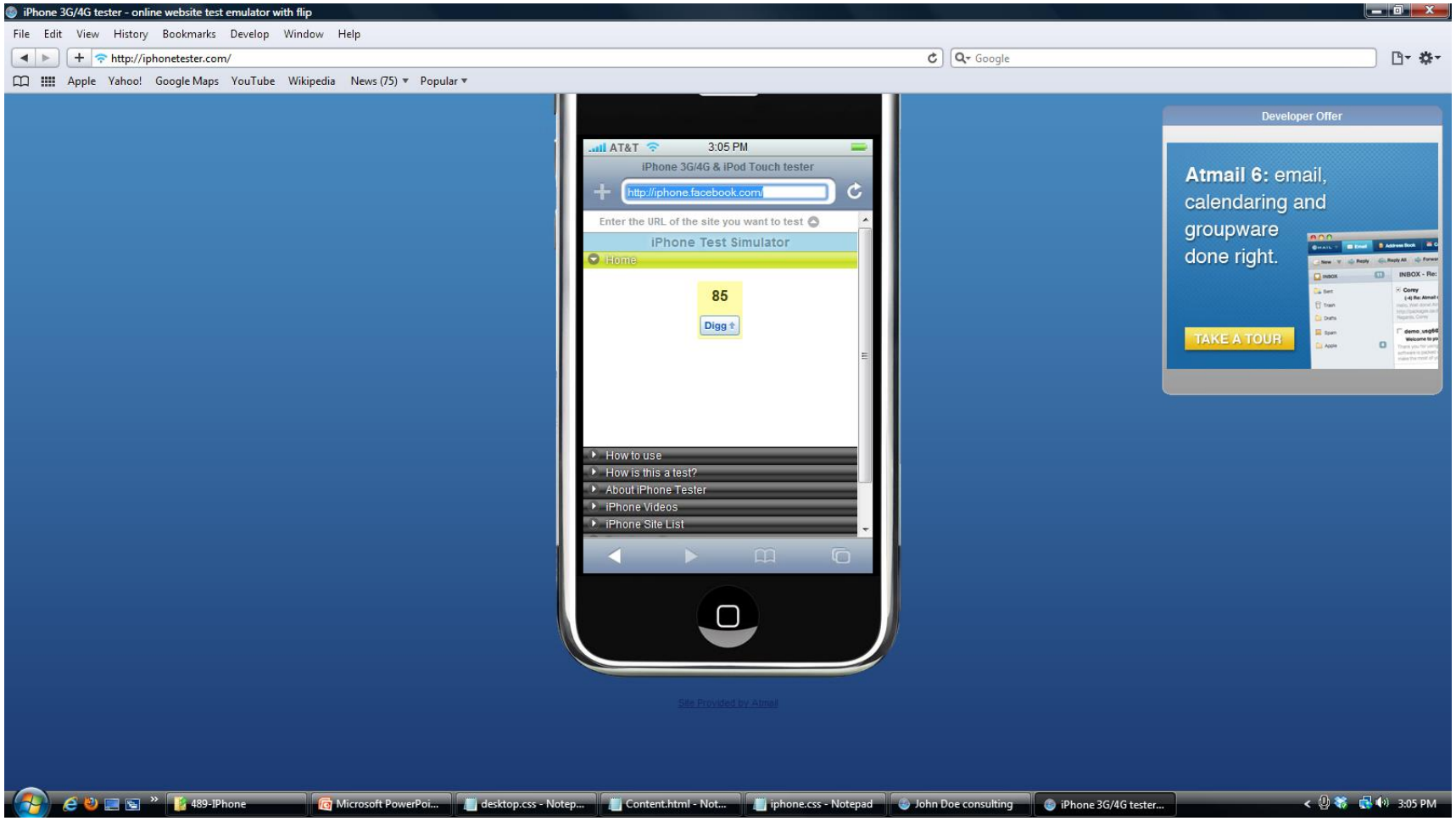
MOBILE HTML 5 compatibility tables

MY BOOKS

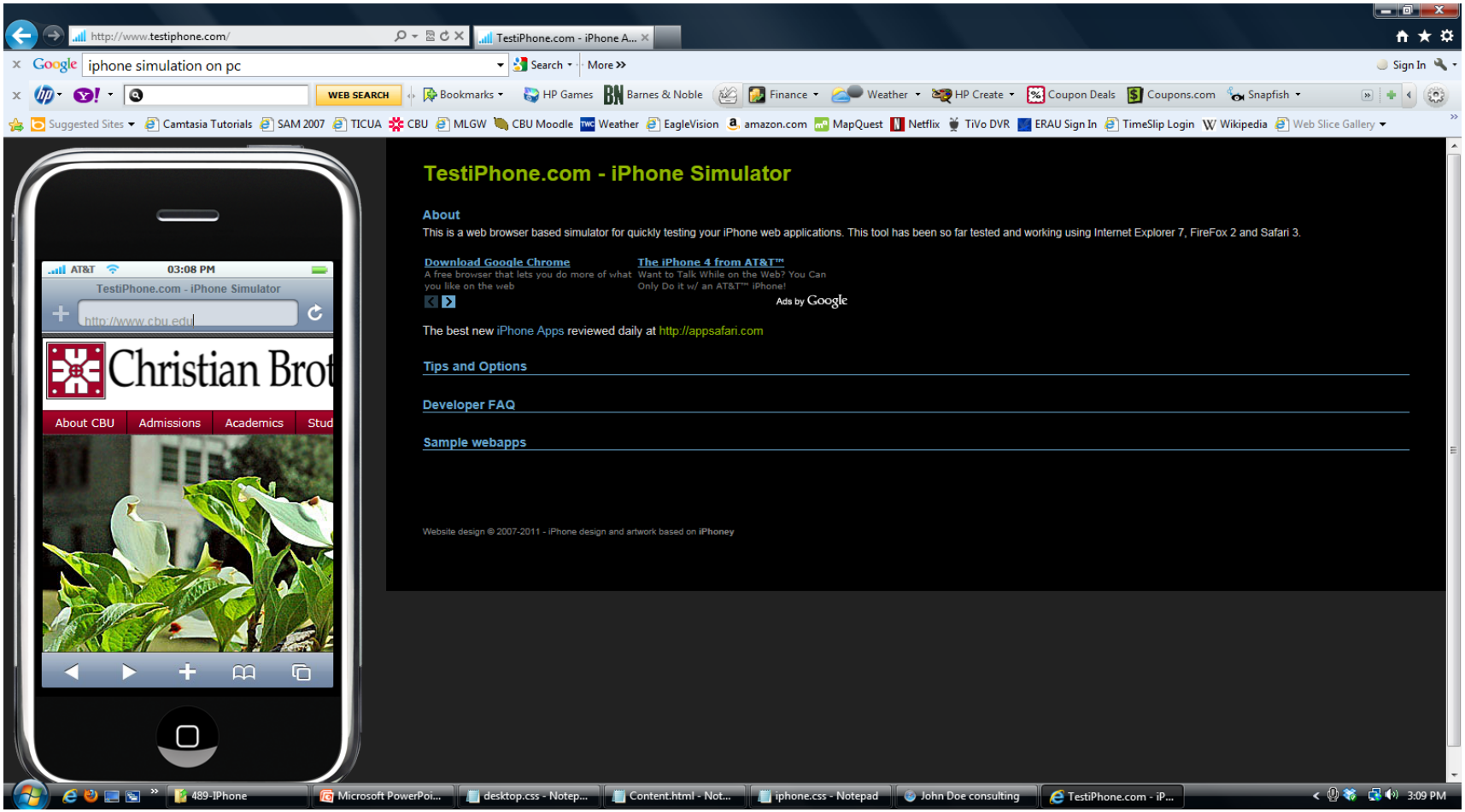


100%

iphonetester.com

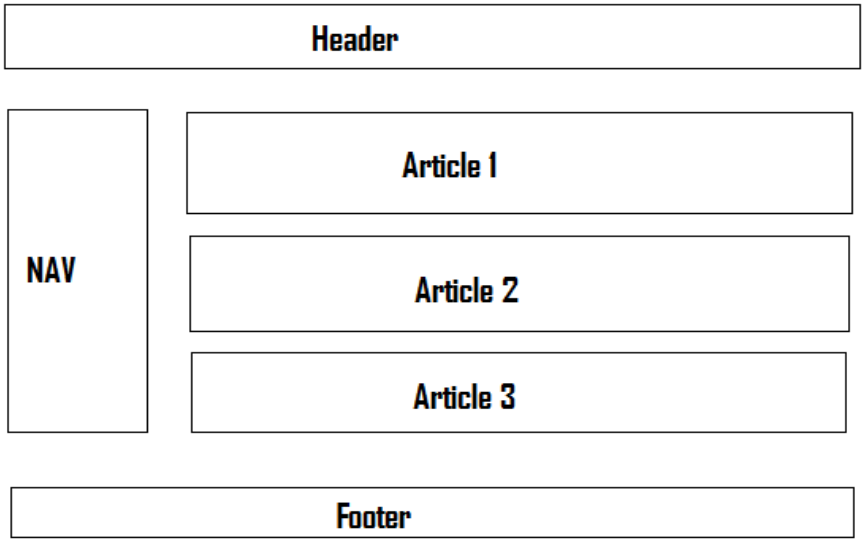


testiphone.com



HTML5 Layout Control – Structural Elements

- HTML5 defines new semantic structural elements that facilitate page layout that traditionally was accomplished with DIV's (and SPAN) but provide more semantics
  - Header
  - Nav
  - Article & Section
  - Footer

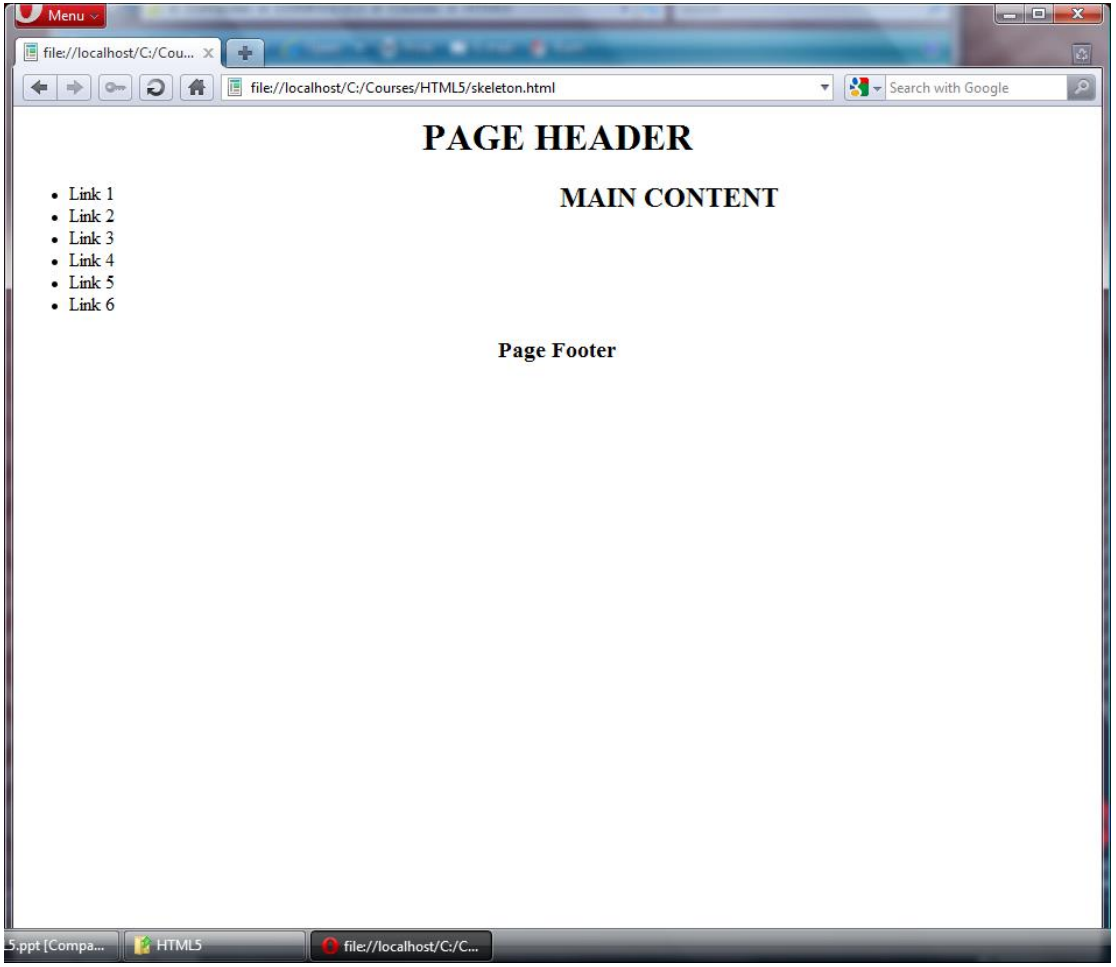


Example HTML5 File

```
<html lang="en">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
</head>
<body>
        <header>
                <h1 align="center">PAGE HEADER</h1>
        </header>
        <nav>
                <ul>
                        <li>Link 1 </li>
                        <li>Link 2 </li>
                        <li>Link 3 </li>
                        <li>Link 4 </li>
                        <li>Link 5 </li>
                        <li>Link 6 </li>
                </ul>
        </nav>
        <article>
                <h2 align="center">MAIN CONTENT</h2>
        </article>
        <footer>
                <h3 align="center">Page Footer</h3>
        </footer>
</body>
</html>
```

Try this !

Example HTML5 File Rendered



CSS Simulation File

- For browsers that do not support these structural tags, one can simulate that using CSS
- The CSS file to “simulate” HTML5 styling is:
  - header, nav, footer, article {display:block}
  - nav {float:left; width:20%}
  - article {float:right; width:79%}
  - footer {clear:both}

Example HTML5 File

```
<html lang="en">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
        <link rel="stylesheet" type="text/css" href="html5_css.css">
</head>
<body>
        <header>
                <h1 align="center">PAGE HEADER</h1>
        </header>
        <nav>
                <ul>
                        <li>Link 1 </li>
                        <li>Link 2 </li>
                        <li>Link 3 </li>
                        <li>Link 4 </li>
                        <li>Link 5 </li>
                        <li>Link 6 </li>
                </ul>
        </nav>
        <article>
                <h2 align="center">MAIN CONTENT</h2>
        </article>
        <footer>
                <h3 align="center">Page Footer</h3>
        </footer>
</body>
</html>
```

IE and CSS Elements

- The previous CSS file to simulate new HTML5 styling will work in all browsers except older IE
- For IE before version 9, the following JavaScript will also need to be added:
  - document.createElement('header');
  - document.createElement('nav');
  - document.createElement('article');
  - document.createElement('footer');
- And JavaScript will need to enabled

SmartPhone App Implementation

- One can build totally separate PC web Apps and Smartphone Apps
  - And then based upon either the screen size or the “user agent” JavaScript variable (or another browser/platform detection mechanism) – switch the URL
- Or one can use the same content for both platforms (PC and Smartphone) and then use CSS and/or JavaScript to just switch the applied style sheets

Working with DOM Objects

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>DOM Example Without jQuery</title>
                <script type="text/javascript">
                        function changeText () {
                          document.getElementById('greeting').innerHTML="Hello !";
                        }
                </script>
        </head>
        <body>
                <h1 id="greeting" onclick="changeText()">Click Here</h1>
        </body>
</html>
```

Using jQuery Library

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>DOM Example Without jQuery</title>
                <script type="text/javascript" src="jquery.js"></script>
                <script type="text/javascript">
                        function changeText () {
                                $('#greeting').text('Hello !');
                        }
                </script>
        </head>
        <body>
                <h1 id="greeting" onclick="changeText()">Click Here</h1>
        </body>
</html>
```

Media Queries or Qualifiers

- Media queries associate a style sheet or style rule with a specific device or list of device features
- Create a media query within an HTML file, by adding a media attribute to either the `link` or `style` element in the document head:

  `media="devices"`

  where `devices` is a comma-separated list of supported media types associated with a specified style sheet

Media Types

Media Type	Used For
`all`	All output devices (the default)
`braille`	Braille tactile feedback devices
`embossed`	Paged Braille printers
`handheld`	Mobile devices with small screens and limited bandwidth
`print`	Printers
`projection`	Projectors
`screen`	Computer screens
`speech`	Speech and sound synthesizers, and aural browsers
`tty`	Fixed-width devices such as teletype machines and terminals
`tv`	Television-type devices with low resolution, color, and scrollability

Media Queries (con't)

- Media queries can be used to associate specific style rules with specific devices using the following:

```
@media devices {
   style rules
}
```

where `devices` are supported media types and `style rules` are the style rules associated with those devices

Media Queries (con't)

- To target a device based on its features, add the feature and its value to the media attribute using the syntax:

  `media="devices and|or (feature:value)"`

  where `feature` is the name of a media feature and `value` is the feature's value
- The `and` and `or` keywords are used to create media queries that involve different devices or features, or combinations of both
- Or once can link to a specific style sheet for that media type

Media Queries (con't)

Feature	Description
`aspect-ratio`	The ratio of the width of the display area to its height
`color`	The number of bits per color component of the output device; if the device does not support color, the value is 0
`color-index`	The number of colors supported by the output device
`device-aspect-ratio`	The ratio of the device-width value to the device-height value
`device-height`	The height of the rendering surface of the output device
`device-width`	The width of the rendering surface of the output device
`height`	The height of the display area of the output device
`monochrome`	The number of bits per pixel in the device's monochrome frame buffer
`orientation`	The general description of the aspect ratio: equal to `portrait` when the height of the display area is greater than the width; equal to `landscape` otherwise
`resolution`	The resolution of the output device in pixels, expressed in either dpi (dots per inch) or dpcm (dots per centimeter)
`width`	The width of the display area of the output device

Stylesheet Media Qualifier

```
<HTML>
  <HEAD>
    <TITLE>Link to a target medium</TITLE>
   <LINK REL="stylesheet" TYPE="text/css“
   MEDIA="print, handheld" HREF=“xxxxxx.css">
  </HEAD>
  <BODY>
    <P>The body...
  </BODY>
</HTML>
```

Media Queries (con't)

- The **mobile first** principle is one in which the overall page design starts with base styles that apply to all devices followed by style rules specific to mobile devices
- Tablet styles are generally applied when the screen width is 481 pixels or greater
- Desktop styles build upon the tablet styles when the screen width exceeds 768 pixels
- As the screen width increases, more features found in smaller devices are added or replaced

Media Queries (con't)

Media Queries (con't)

- <link rel="stylesheet" media="screen and (min-device-width: 800px)" href="800.css" />
- The above code will apply the 800.css styling to the document only if the device viewing it has a width of 800px or wider
- This is the device width, not the current width of the browser window
  - On the iPhone that means 480px (or 980px for iPad)
  - MacBook Pro is going to return 1920px for the device width
- The actual browser window may be a portion of that at any moment
- The device width is quite useful when dealing with mobile devices where the browser is probably 100% of the screen whenever in use, but less useful in laptop/desktop browsers

Viewport

- Mobile browsers may render pages in a virtual "window" (the viewport), usually wider than the screen, so they don't need to squeeze every page layout into a tiny window (which would break many non-mobile-optimized sites) - users can pan and zoom to see different areas of the page
- Mobile Safari introduced the "viewport meta tag" to let web developers control the viewport's size and scale; many other mobile browsers now support this tag; if not supported, it is just ignored

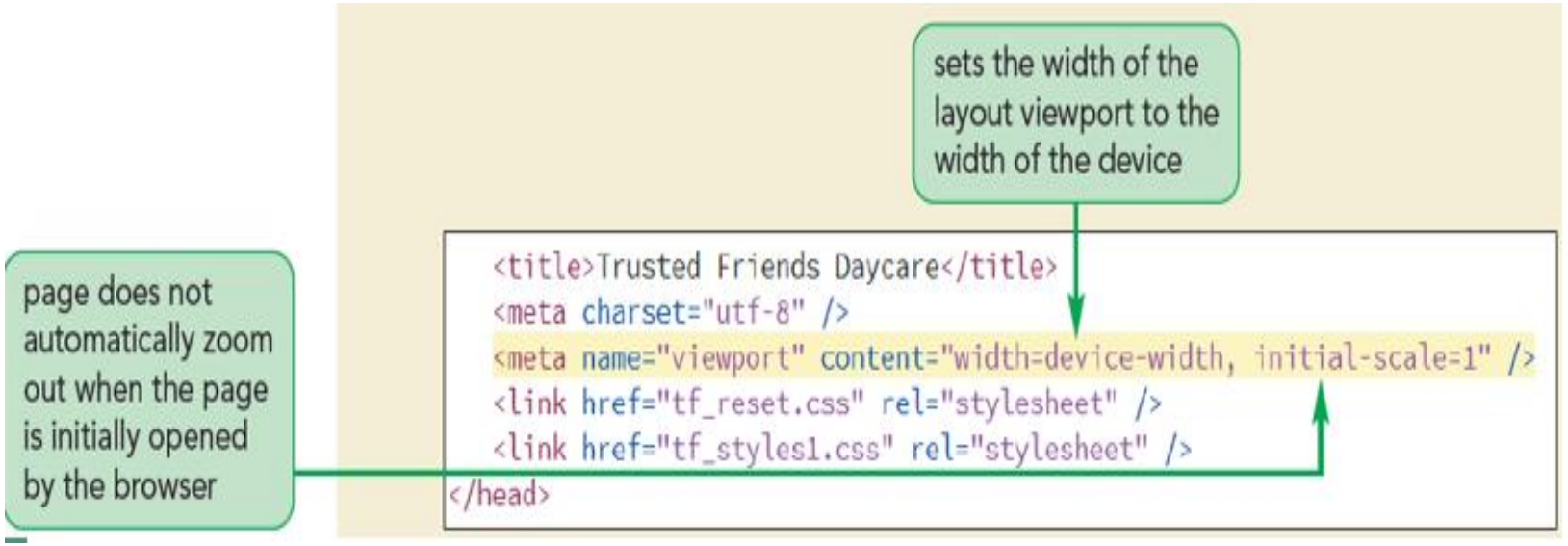
Without the viewport meta tag

With the viewport meta tag

Viewport (con't)

- <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
- The initial-scale property controls the zoom level when the page is first loaded, the maximum-scale, minimum-scale, and user-scalable properties control how users are allowed to zoom the page in or out
- Mobile devices have two types of viewports:
  - **Visual viewport** – displays the web page content that fits within a mobile screen
  - **Layout viewport** – contains the entire content of the page, some of which may be hidden from the user

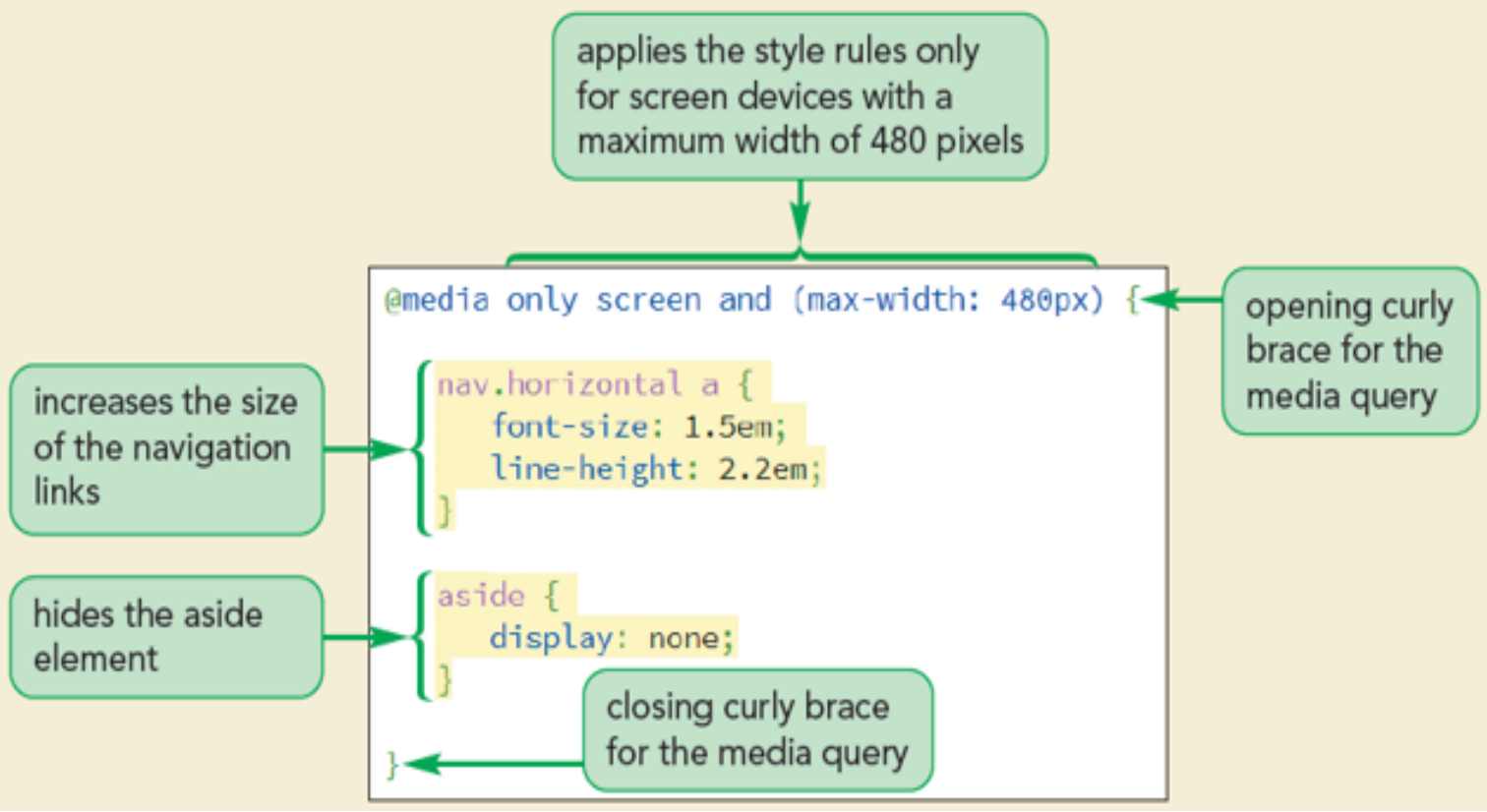
Viewport (con't)



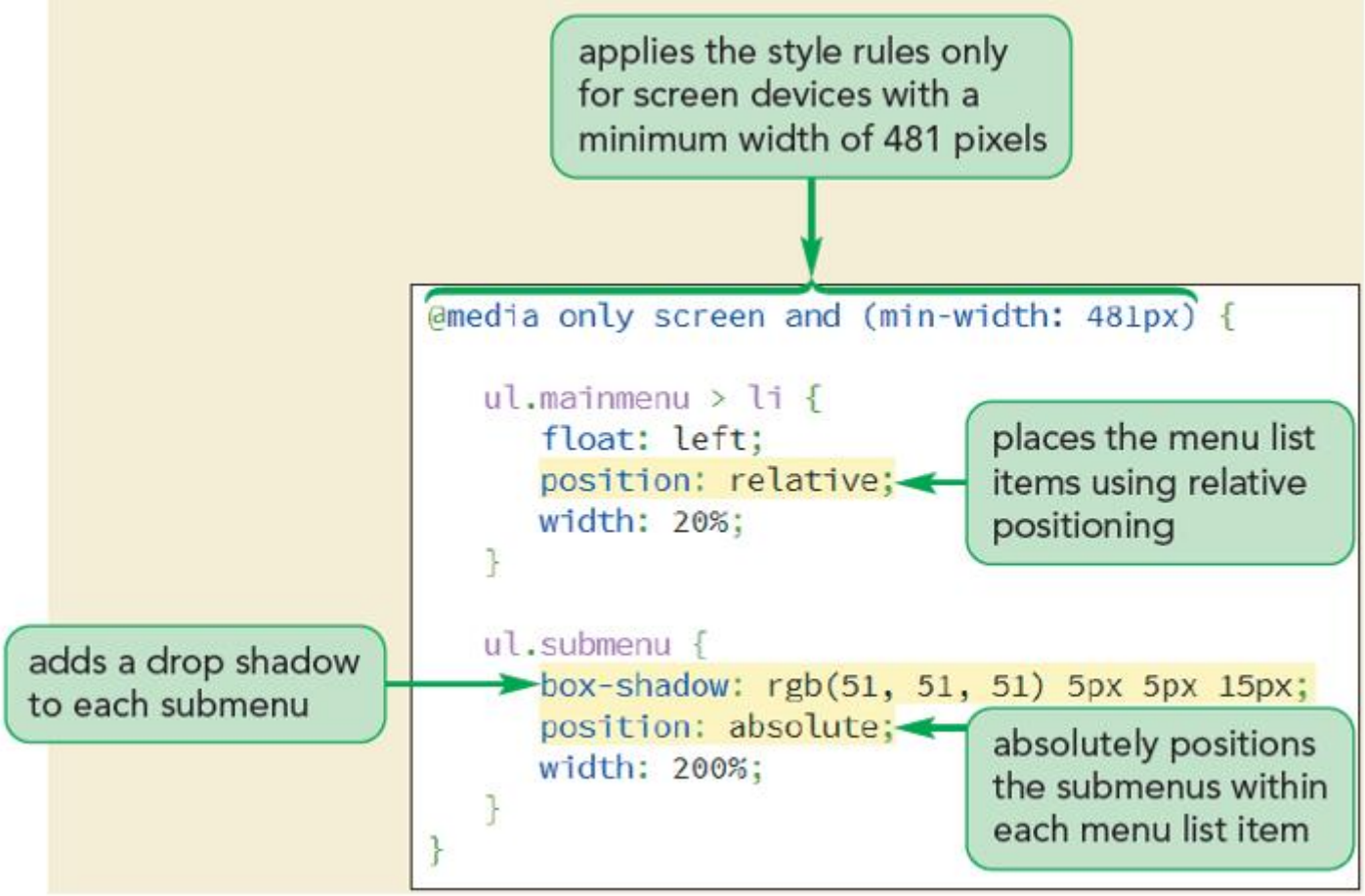
Viewport (con’t)

- We can also specify stylesheets that are only to be used when the viewport is between two different pixel sizes:
  - <link rel='stylesheet' media='screen and (min-width: 701px) and (max-width: 900px)' href='css/medium.css' />
- This stylesheet will only take affect when the current browser window is between 701 and 900 pixels in width

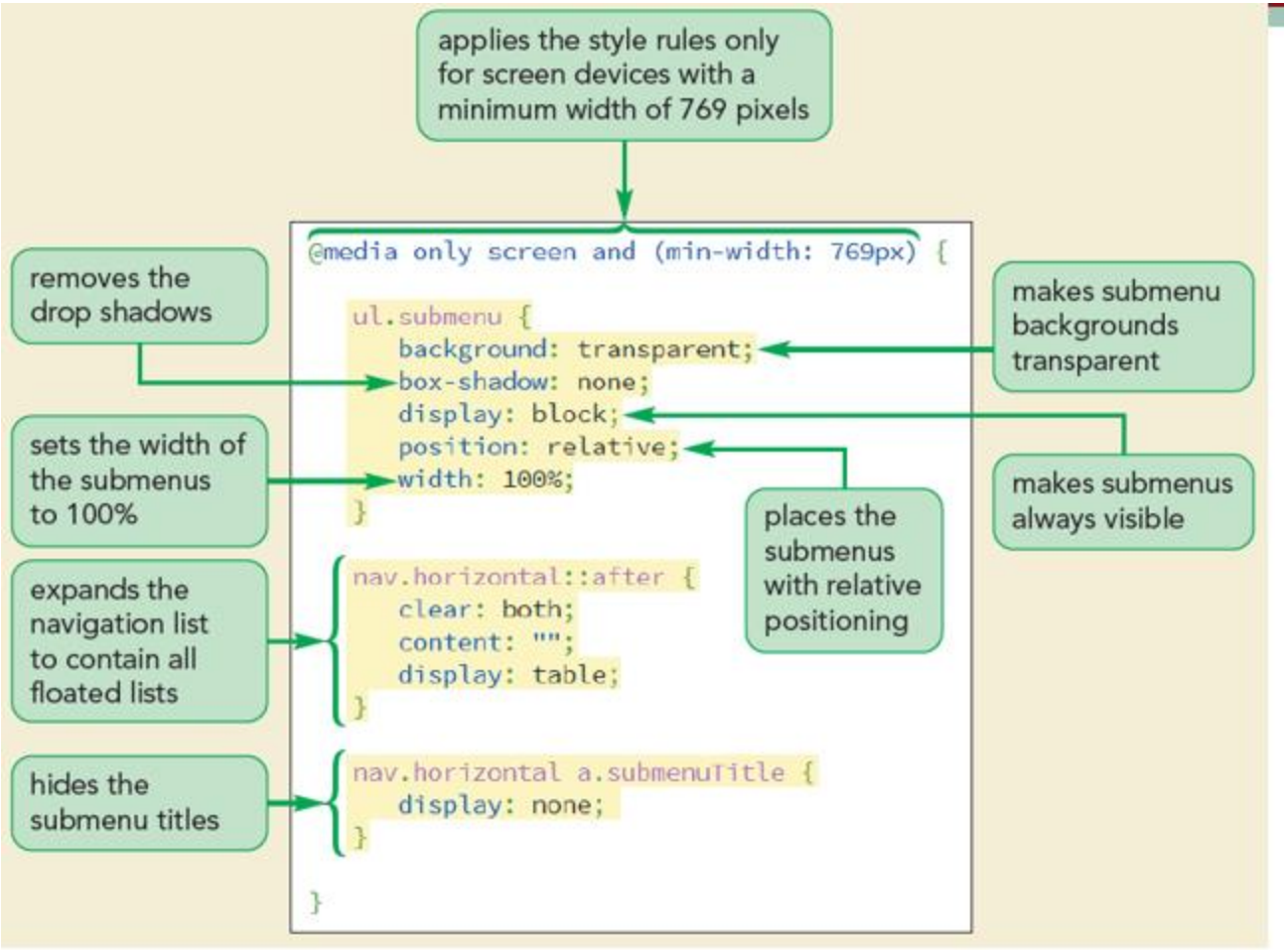
Example of Mobile Styles



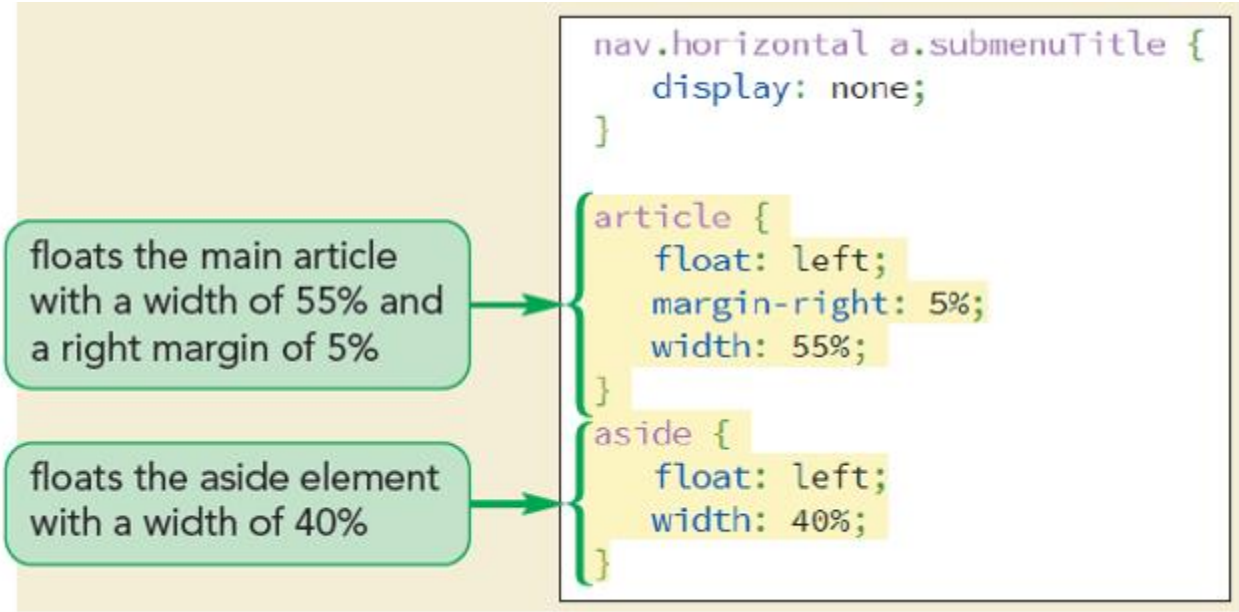
Example of Tablet Styles



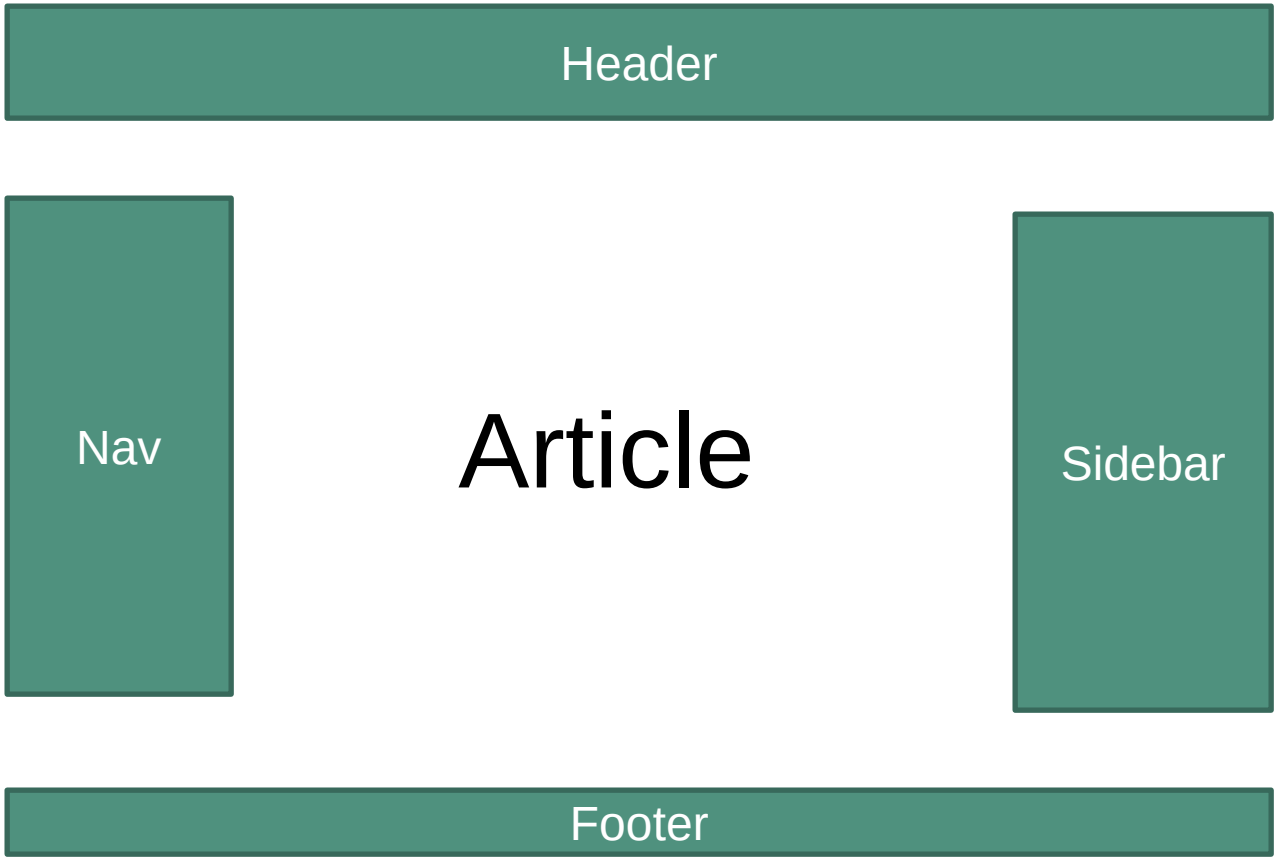
Example of Desktop Styles



Example of Desktop Styles (con't)



Smartphone Methodology Based Upon Generic Page Design



Smartphone Viewport & Stylesheets

- Meta tag to tell the iPhone browser (Safari) that viewport is only 480 (otherwise Safari would assume 980, and page would be loaded to small)
  - <meta name="viewport" content="user-scalable=no, width=device-width" />
- Separate stylesheets for desktop and iPhone:
  - <link rel="stylesheet" type="text/css" href="iphone.css" media="only screen and (max-width: 480px)" />
  - <link rel="stylesheet" type="text/css" href="desktop.css" media="screen and (min-width: 481px)" />

Example Page HTML

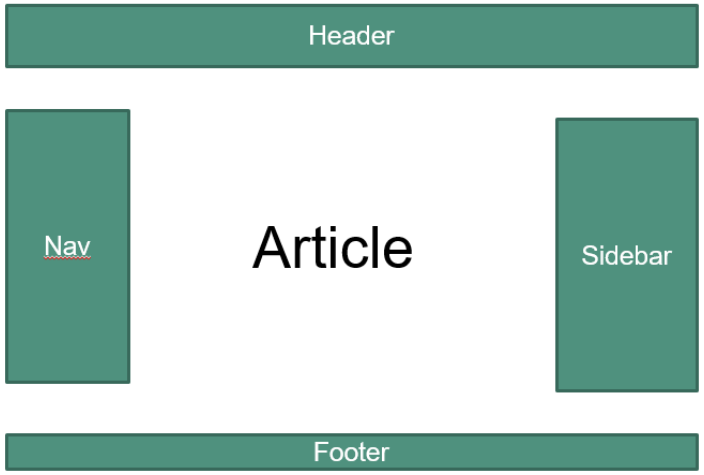
```
<!DOCTYPE html>
<html>
  <head>
    <title>John Doe Consulting</title>
                <meta charset="utf-8">
    <meta name="viewport" content="user-scalable=no, width=device-width" />
    <link rel="stylesheet" type="text/css" href="iphone.css" media="only screen and (max-width: 480px)" />
    <link rel="stylesheet" type="text/css" href="desktop.css" media="screen and (min-width: 481px)" />
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript" src="iphone.js"></script>
  </head>
  <body>
            <container>
      <header>
                                        <h1 align="center">John Doe Consulting</h1>
          </header>
      <nav>
        <ul>
          <li><a href="consulting.html">Consulting</a></li>
          <li><a href="support.html">Support</a></li>
          <li><a href="development.html">Development</a></li>
                                                    <li><a href="client.html">Client List</a></li>
                                                    <li><a href="testa.html">Testamonials</a></li>
                                                    <li><a href="product.html">Products</a></li>
        </ul>
      </nav>
      <article>
                                        <h2 align="center">About John Doe</h2>
        <p align="center">John Doe is a web developer and consultant. His consulting firm, John Doe Consulting, Inc., has been developing successful web sites in many industries for over 10 years, including mobile applications.</p>
          </article>
      <sidebar>
        <p style=text-align:center><img alt="Photo of John Doe" src="smile.jpg"></p>
        <p>Jonh Doe Contact Info</p>
                                        <ul>
          <li>Phone: 999-111-2222</li>
          <li>Mobile: 888-222-3333</li>
          <li>Email: jdoe12409@yahoo.com</li>
                                        </ul>
          </sidebar>
      <footer>
        <table width=100%>
                                          <tr>
          <td align="center" width="33%"><a href="services.html">Services</a></td>
          <td align="center" width="33%"><a href="about.html">About</a></td>
          <td align="center"><a href="blog.html">Blog</a></td>
                                          </tr>
            </table>
        <p class="subtle" align="center">John Doe Consulting, Inc.</p>
      </footer>
    </container>
  </body>
</html>
```

Example Page HTML - Header

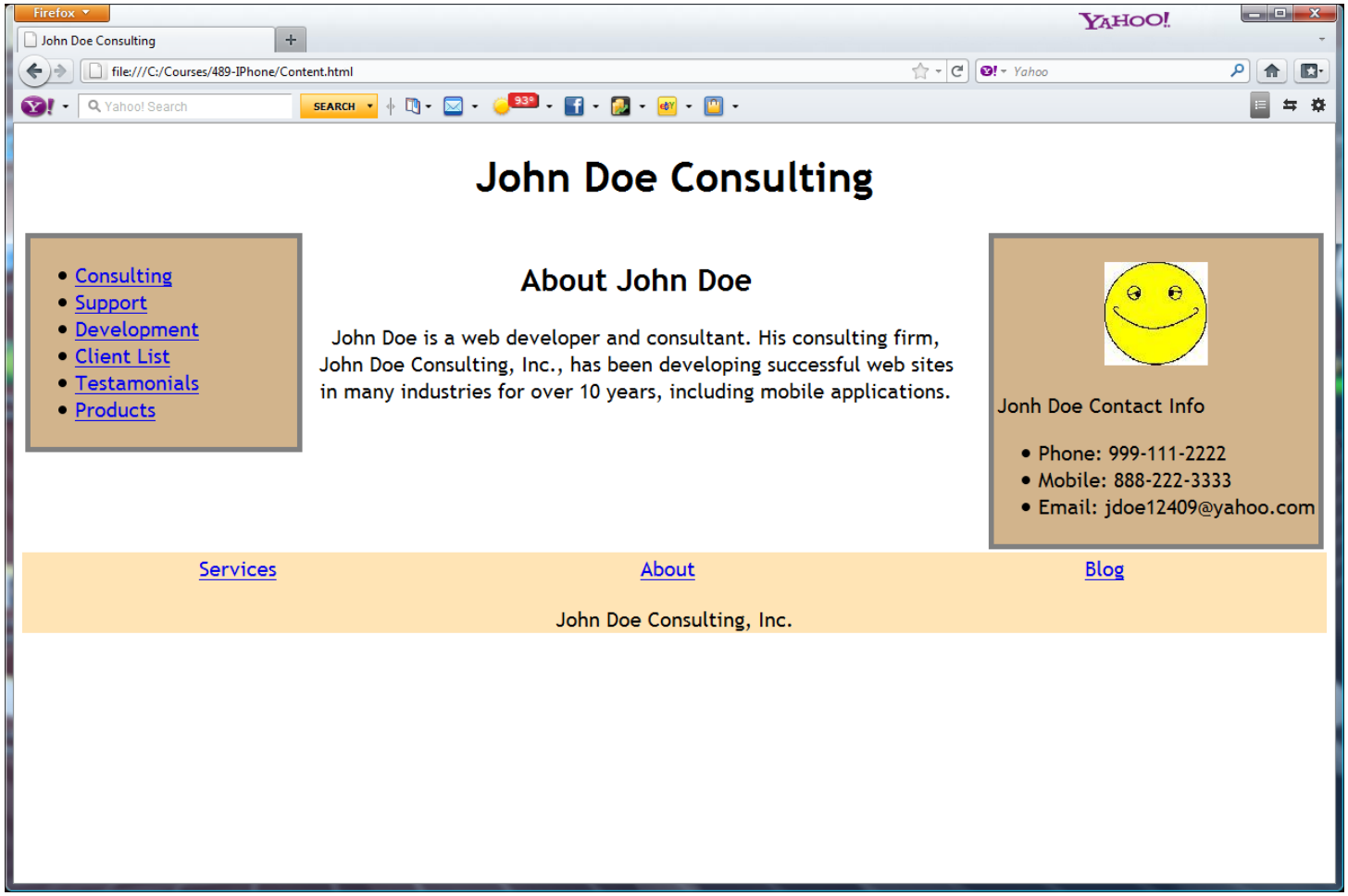
```
<head>
    <title>John Doe Consulting</title>
     <meta charset="utf-8">
    <meta name="viewport" content="user-scalable=no, width=device-width" />
    <link rel="stylesheet" type="text/css" href="iphone.css" media="only screen and
                (max-width: 480px)" />
    <link rel="stylesheet" type="text/css" href="desktop.css" media="screen and
                (min-width: 481px)" />
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript" src="iphone.js"></script>
  </head>
```

Desktop Stylesheet

- body {font: 125% "Lucida Grande", "Trebuchet MS", Verdana, sans-serif;}
- header, nav, footer, article, sidebar {display:block;}
- nav {float:left; width:20%; border:5px solid gray; background-color:tan; padding:3px; margin:3px;}
- sidebar {float:right; border:5px solid gray; background-color:tan; padding:3px; margin:3px;}
- article {float:left; width:50%; padding:2px; margin:2px;}
- footer {clear:both;background-color:moccasin;}



Rendered Page on Desktop



iPhone Stylesheet – first cut

- body {background-color:#ddd; color:#222; font-family:Helvetica; font-size:14px; margin:0; padding:0; }
- nav {padding: 5px;}
- article {padding:5px;}
- sidebar {padding:5px;}
- footer {display:none;}

Rendered Page on iPhone

Improving Smartphone Look & Feel

- Buttons to expand/contract detail
- Aesthetics
- Detecting user agent, etc.
- Dynamically changing styles
- Using other smartphone capabilities

JavaScript for Button (iPhone.js)

```
if (window.innerWidth && window.innerWidth <= 480 {
    $(document).ready(function(){
        $('nav ul').addClass('hide');
        $('header').append('<div class="leftButton"
onclick="toggleMenu()">Menu</div>');
    });
    function toggleMenu() {
        $('nav ul').toggleClass('hide');
        $('header .leftButton').toggleClass('pressed');
    }
}
```

Webkit


The WebKit Open Source Project
Welcome to the website for the WebKit Open Source Project!
WebKit is an open source web browser engine. WebKit is also the name of the Mac OS X system framework version of the engine that's used by Safari, Dashboard, Mail, and many other OS X applications. WebKit's HTML and JavaScript code began as a branch of the KHTML and KJS libraries from KDE.
Getting involved
There are many ways to get involved. You can:
download the latest nightly build
install developer tools and then check out and build the source code
Download
Nightly builds
Once you have either of these, you can help by:
reporting bugs you find in the software
providing reductions to bugs
submitting patches for review
More info
More information about WebKit can be found on its wiki. You can help here too, by adding information that can help others learn about WebKit. If you have more questions, contact us.
Projects
There are many exciting (new) projects that you can contribute to:
help us improve Website compatibility
write documentation
SVG
MathML
CSS
DOM
Home
Surfin' Safari Blog
Planet WebKit
Project Goals
Keeping in Touch
Trac
Contributors Meeting
Working with the Code
Installing Developer Tools
Getting the Code
Building WebKit
Running WebKit
Debugging WebKit
Contributing Code
Commit and Review Policy
Adding Features
Security Policy
Documentation
Wiki
Projects
Code Style Guidelines
Technical Articles
Web Inspector
Web Developer Resources
Demos
Testing
Regression Testing
Leak Hunting
Writing New Tests
Getting a Crash Log
Bugs
Reporting Bugs
Bug Report Guidelines
Bug Prioritization
Test Case Reduction
Bug Life Cycle
WebKit is open source software with portions licensed under the LGPL and

http://qooxdoo.org/documentation/general/webkit_css_styles

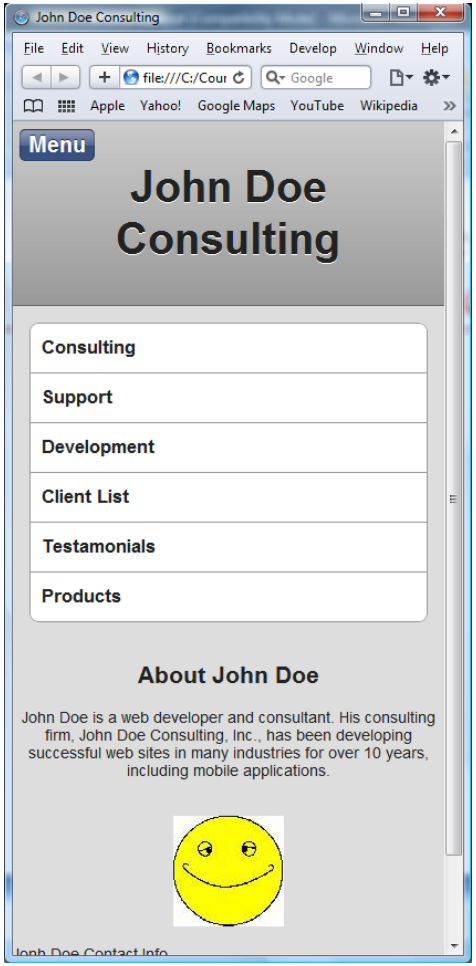
iPhone Stylesheet – polished
[making an iPhone App look like one]

- body {background-color:#ddd; color:#222; font-family:Helvetica; font-size:14px; margin:0; padding:0; }
- header {background-color:#ccc; border-bottom:1px solid #666; color:#222; display:block; font-size:20px; font-weight:bold;padding:10px 0; text-align:center; text-decoration:none;}
- header {text-shadow:0px 1px 0px #fff; background-image:-webkit-gradient(linear, left top, left bottom, from(#ccc), to(#999));}
- header div.leftButton {position: absolute; top:7px; left:6px; height: 30px; font-weight:bold; text-align:center; color:white; text-shadow:rgba(0,0,0,0.6) 0px -1px 0px; line-height:28px; border-width:0 8px 0 8px; -webkit-border-image:url(button.png) 0 8 0 8;}
- header div.pressed {-webkit-border-image:url(button_clicked.png) 0 8 0 8;}
- nav {padding:5px;}
- nav ul {list-style:none; margin:10px; padding:0;}
- nav ul li a {background-color:#FFFFFF; border:1px solid #999999; color:#222222; display:block; font-size:17px; font-weight:bold; margin-bottom:-1px; padding:12px 10px; text-decoration:none;}
- nav ul li:first-child a {-webkit-border-top-left-radius:8px; -webkit-border-top-right-radius:8px;}
- nav ul li:last-child a {-webkit-border-bottom-left-radius:8px; -webkit-border-bottom-right-radius:8px;}
- nav ul.hide {display: none;}
- article {padding:5px;}
- sidebar {padding:5px;}
- footer a {background-color:#FFFFFF; border:1px solid #999999; color:#222222; display:block; font-size:17px; font-weight:bold; margin-bottom:-1px; padding:12px 10px; text-decoration:none;}

Rendered Page on iPhone

Rendered Page on iPhone

[after hitting menu button]



Checking User Agent

- Use a smartphone detection JavaScript library
- Or code one yourself:
  - var deviceIphone = "iphone";
  - var deviceIpod = "ipod";
  - var uagent = navigator.userAgent.toLowerCase();
  - function DetectIphone() {
    - if (uagent.search(deviceIphone) > -1) return true; else return false; }
  - function DetectIpod() {
    - if (uagent.search(deviceIpod) > -1) return true; else return false; }
  - function DetectIphoneOrIpod() {
    - if (DetectIphone()) return true; else if (DetectIpod()) return true; else return false; }
- Then use in iPhone.js
  - if (window.innerWidth && window.innerWidth <= 480 || DetectSmartphone() == true) {

Dynamically Switching Style Sheets

```
function loadjscssfile (filename, filetype){
 if (filetype=="js"){ //if filename is a external JavaScript file
  var fileref=document.createElement('script')
  fileref.setAttribute("type","text/javascript")
  fileref.setAttribute("src", filename)
 }
 else if (filetype=="css"){ //if filename is an external CSS file
  var fileref=document.createElement("link")
  fileref.setAttribute("rel", "stylesheet")
  fileref.setAttribute("type", "text/css")
  fileref.setAttribute("href", filename)
 }
 if (typeof fileref!="undefined")
  document.getElementsByTagName("head")[0].appendChild(fileref)
}

function removejscssfile (filename, filetype){
 var targetelement=(filetype=="js")? "script" : (filetype=="css")? "link" : "none" //determine element type to create nodelist from
 var targetattr=(filetype=="js")? "src" : (filetype=="css")? "href" : "none" //determine corresponding attribute to test for
 var allsuspects=document.getElementsByTagName(targetelement)
 for (var i=allsuspects.length; i>=0; i--){ //search backwards within nodelist for matching elements to remove
  if (allsuspects[i] && allsuspects[i].getAttribute(targetattr)!=null && allsuspects[i].getAttribute(targetattr).indexOf(filename)!=-1)
   allsuspects[i].parentNode.removeChild(allsuspects[i]) //remove element by calling parentNode.removeChild()
 }
}
```

Dynamically Switching Style Sheets

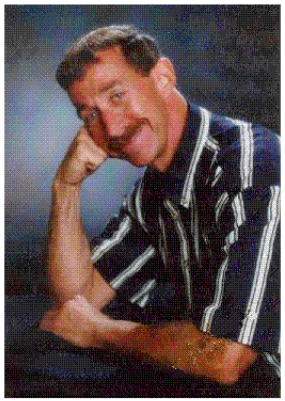
```
if (window.innerWidth && window.innerWidth <= 480 || DetectSmartphone() == true) {
        loadjscssfile("iphone.css", "css")
…
}else {
        loadjscssfile("desktop.css", "css")
}
window.onresize = function () {
        if (window.innerWidth && window.innerWidth <= 480 || DetectSmartphone() == true) {
                removejscssfile("desktop.css", "css");
                loadjscssfile("iphone.css", "css")
                …               }
        else {
                removejscssfile("iphone.css", "css")
                loadjscssfile("desktop.css", "css")
                …
        }
}
```

My Website in Browser

Dan's Home Page

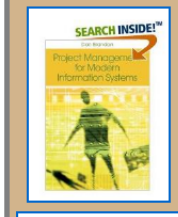
Courses

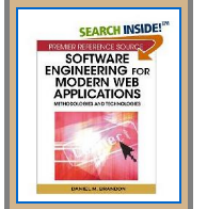
- Computer Business Apps
- Introduction to MIS
- Intro to Computer Sci
- Elementary Statistics
- Intermediate Statistics
- C/C++ Programming
- Advanced Excel & VBA
- Object Oriented Design
- Advanced C++
- Web Programminmg
- Database Design
- Java Programming
- Decision Support
- Info Systems Mgmt
- MBA Info Systems Mgmt
- Intro to Project Mgmt
- Advanced Project Mgmt
- Research Methods



Contact Info
Biography
Academic Interest
Publications
Course Policies
MIS Department
School of Business
CBU Home Page

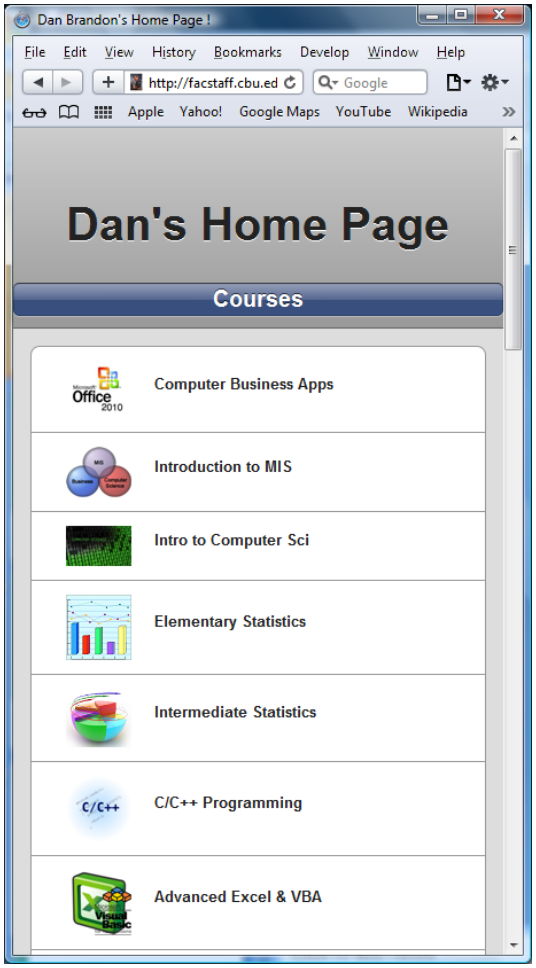
New Books



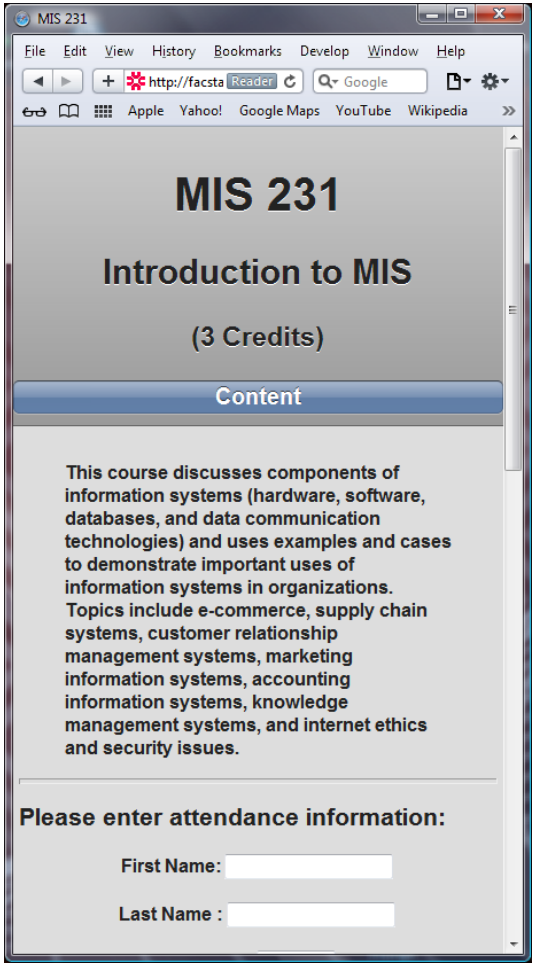


My Website in SmartPhone [facstaff.cbu.edu/dbrandon]

My Website in SmartPhone
[after hitting "courses" button]

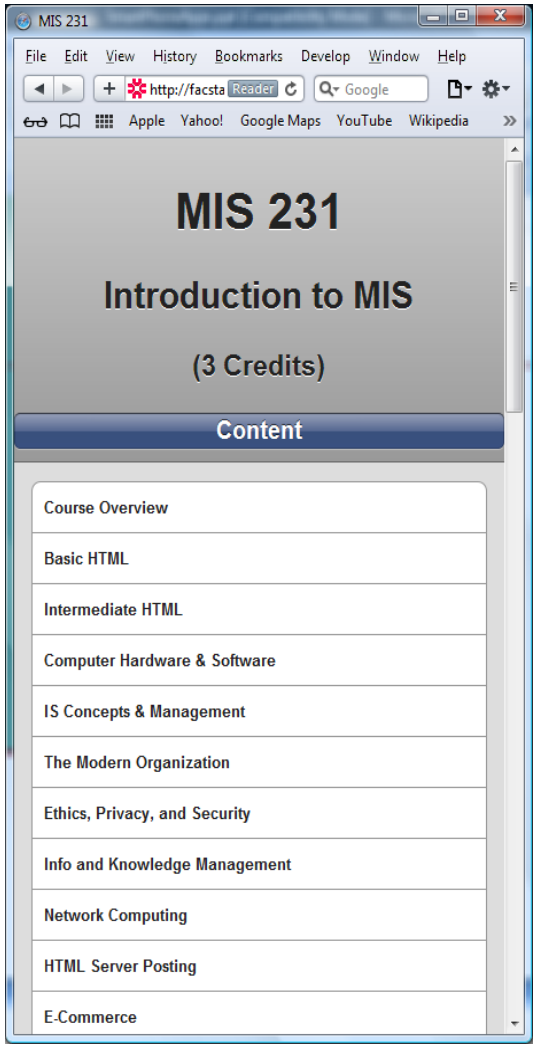


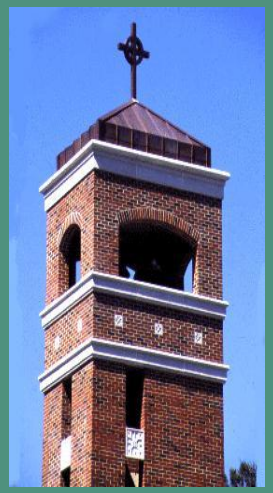
Course Syllabus



Course Syllabus

[after expanding "content" button]





jQuery Mobile

Dan Brandon, Ph.D., PMP

Christian Brothers University

Memphis, TN USA

jQuery Mobile (jQM)

jQuery Mobile Definition

- A unified user interface system across all popular mobile device platforms
- Built on jQuery and jQuery UI foundation
- Flexible and easy themeable design

jQuery Mobile Features

- “Canned” but parameter driven CSS and Javascript with image resources
- Automatic AJAX
- Progressive enhancement (graceful degradation for less capable browsers)
- Accessibility support (where such support is provided by underlying browser and OS such as in iOS)
  - iOS – Setings→General→Accessibility→Activate VoiceOver
- Works with PhoneGap (converts WebApp into native App)
- Supported by latest Dreamweaver release

jQuery Mobile Features (con't)

- Relatively platform independent for smartphones and tablets (targets "touch" devices)
  - 3000+ mobile devices
  - 50+ browsers
- Best for smartphone/tablet apps only – **not** content to be delivered to both traditional desktops and smartphones
- Allows use of basic smartphone features →
- Not for all mobile apps such as games

SmartPhones and Tablets

[some mobile smart devices do not have phones]

- Multitasking OS
- HTML5 Browser
- Wireless LAN (WLAN or WiFi)
- 3G and/or 4G Connection
- Touch Screen
- Music Player
- GPS
- Digital Compass
- Still and Video Camera
- Bluetooth
- Accelerometer
- Gyroscope

Types of Mobile WebApps

- Accessed from the mobile device web browser
- Installed as a full-screen webapp
- As an installed webabb via a package officially implemented by vendors (“widget”)
- As an installed webapp embedded in a native application (“hybrid”) – typically via PhoneGap

jQM Required Resources

- Resource options:
  - Download and host on your server (necessary if using PhoneGap)
  - Link to content delivery network (code.jquery.com)
- Resource files:
  - jQuery core JavaScript file
  - jQuery Mobile core JavaScript file
  - jQuery Mobile core CSS file
  - jQuery Mobile theme CSS file (optional)

jQuery and jQuery Mobile File Links

["min" versions have whitespace removed]

- <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.css" />
- <script
  - src="http://code.jquery.com/jquery-1.6.4.min.js">
- </script>
- <script
  - src="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.js">
- </script>

*** These may not be latest versions. ***

jQM Architecture Concepts

- Role – the architectural purpose of an HTML div
  - {HTML5 custom data attributes}
- Page – a visible screen of content; in jQM a "div" element with the specific "role" of a page; an HTML file can contain multiple pages
- Themes – defines visual appearance; a group of definitions for layout, styles, and colors
  - There is a default theme, and one can create their own custom themes
  - Themes have multiple color swatch options, defined by a letter

Default Color Swatch Designations

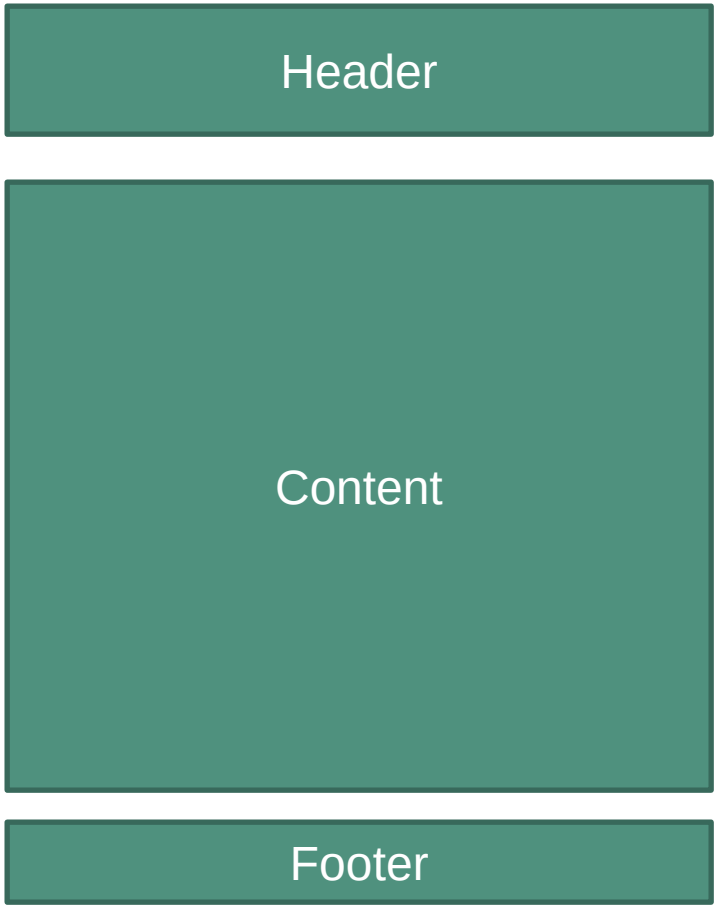
- a – black
  - Highest level of visual priority
- b – blue
  - Secondary level of priority
- c – silver
  - Baseline level
- d – gray
  - Alternate secondary level
- e – yellow
  - Accent

jQM DIV Roles

- page
- header
- footer
- content
- navbar
- button
- listview
- listdivider
- controlgroup
- collapsible
- collapsible-set
- fieldcontain
- dialog
- slider
- nojs

Example jQM Simple Page Layout

[header has left, right, and center areas; with left and right reserved for buttons]



HTML5 jQuery Mobile Skeleton File

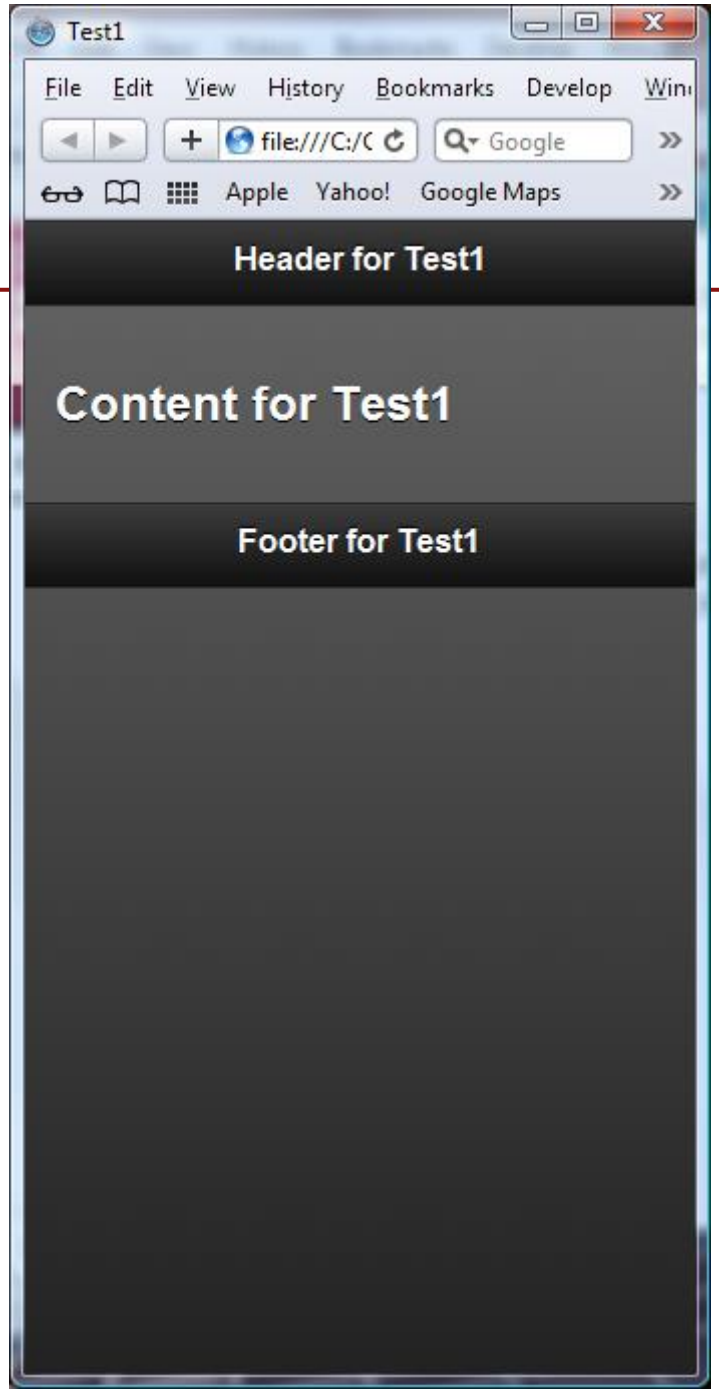
[viewport with scale of 1, so that page fits exactly onto device visible area]

```
<!DOCTYPE html>
<HTML>
<HEAD>
        <meta charset="utf-8">
        <title>…</title>
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.css" />
        <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
        <script src="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.js"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1">
</HEAD>
<BODY>
<div data-role="page" data-theme="a">
        <div data-role="header">…</div>

        <div data-role="content">…</div>

        <div data-role="footer">…</div>
</div>
</BODY>
</HTML>
```

Trivial jQM Example

[only BODY shown]

```
<BODY>
<div data-role="page" data-theme="a">
        <div data-role="header">
                <h1>Header for Test1</H1>
        </div>
        <div data-role="content">
                <h2>Content for Test1</H2>
        </div>
<div data-role="footer">
                <h2>Footer for Test1</H2>
        </div>
</div>
</BODY>
```


Test1
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Header for Test1
Content for Test1
Footer for Test1

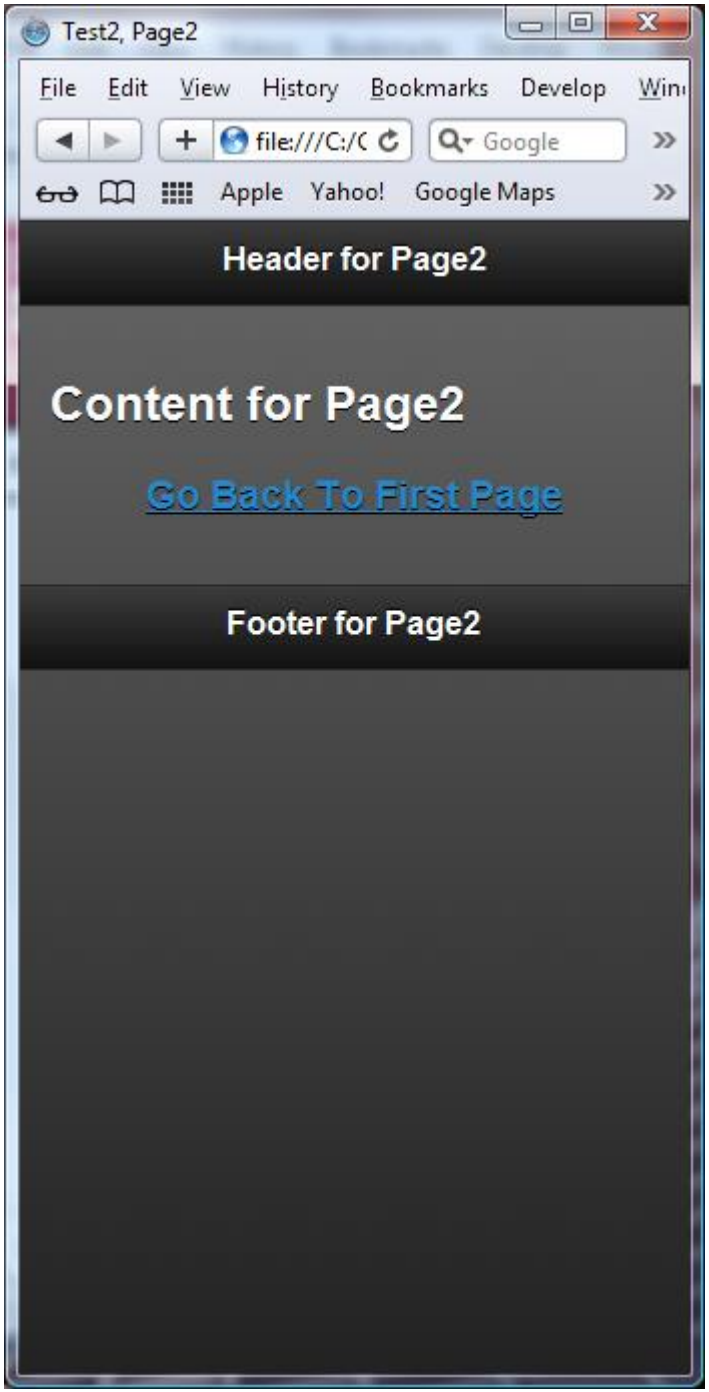
jQM Linking

- Linking types (the first two use AJAX):
  - Link to another page in same HTML file
  - Link to page in another jQM HTML file in the same domain
  - Links to non jQM HTML files or files outside of domain
  - Mobile special links (SmartPhone integration)
- jQM automatically handles linking “transitions”
  - Default is “slide” – others are up, down, pop, fade, and flip
- Also works with browser’s and SmartPhone’s back buttons

Two Page jQM HTML File with Linking

```
<BODY>
<div data-role="page" data-theme="a" id="page1">
        <div data-role="header">
                <h1>Header for Page1</H1>
        </div>
        <div data-role="content">
                <h2>Content for Page1</H2>
                <h3 align="center"><a href ="#page2">Go To Second Page</a></h3>
        </div>
<div data-role="footer">
                <h2>Footer for Page1</H2>
        </div>
</div>
<div data-role="page" data-theme="a" id="page2“ data-title="Test2, Page2">
        <div data-role="header">
                <h1>Header for Page2</H1>
        </div>
        <div data-role="content">
                <h2>Content for Page2</H2>
                <h3 align="center"><a href ="#page1">Go Back To First Page</a></h3>
        </div>
<div data-role="footer">
                <h2>Footer for Page2</H2>
        </div>
</div>
</BODY>
```

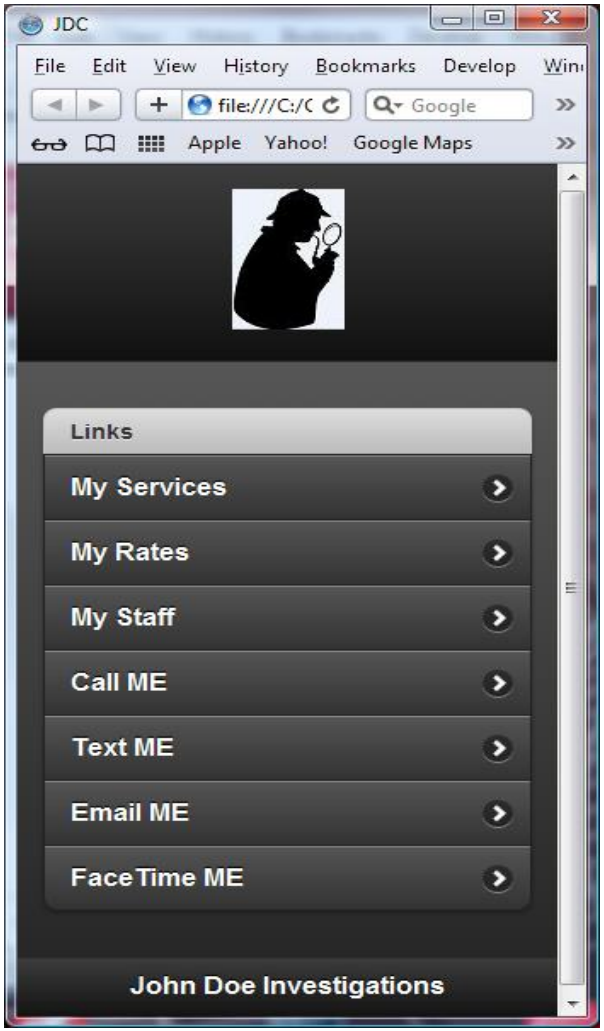

Test2
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Header for Page1
Content for Page1
Go To Second Page
Footer for Page1


Test2, Page2
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Header for Page2
Content for Page2
Go Back To First Page
Footer for Page2

Other Comon jQM Constructs

- Dialog page (modal pop up)
- Lists & separators
- Images and thumbnails
- Form elements
  - Buttons, textbox, textarea, checkbox, radio buttons, menus, sliders, labels, toggles, file upload, etc.
- Toolbars
- Navigation Bars
- Columns and Grids
- Buttons (inline, grouped)
- Icons

Joe Doe Investigations



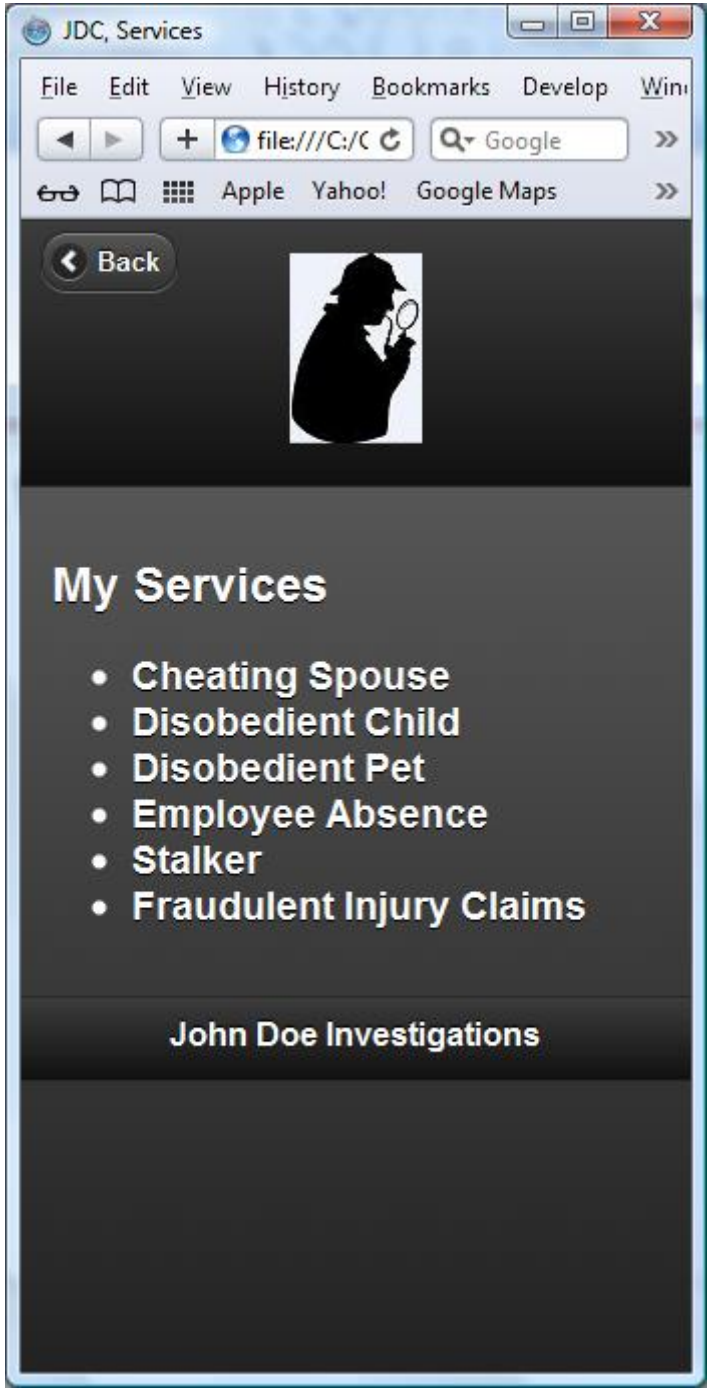

```
<BODY>
<div data-role="page" data-theme="a" id="page1">
                <div data-role="header">
                    <p style=text-align:center><img src="JDC2.jpg" alt="JDC Logo">
                </div>
                <div data-role="content">
                  <ul data-role="listview" data-inset="true" data-divider-theme="d">
                    <li data-role="list-divider">Links</li>
                      <li><a href="#page2">My Services</a></li>
                     <li><a href="#page3">My Rates</a></li>
                     <li><a href="#page4">My Staff</a></li>
                     <li><a href="tel:+19011234567">Call ME</a></li>
                     <li><a href="sms://+19011234567">Text ME</a></li>
                     <li><a href="mailto:jdc@big-server.com">Email ME</a></li>
                     <li><a href="facetime://109999">FaceTime ME</a></li>
                  </ul>
                </div>
                <div data-role="footer">
                                <h4>John Doe Investigations</h4>
                </div>
</div>
```

Joe Doe Investigations – Page 1

```
<BODY>
<div data-role="page" data-theme="a" id="page1">
        <div data-role="header">
                <p style=text-align:center><img src="JDC2.jpg" alt="JDC Logo">
        </div>
        <div data-role="content">
                <ul data-role="listview" data-inset="true" data-divider-theme="d">
                        <li data-role="list-divider">Links</li>
                        <li><a href="#page2">My Services</a></li>
                        <li><a href="#page3">My Rates</a></li>
                        <li><a href="#page4">My Staff</a></li>
                        <li><a href="tel:+19011234567">Call ME</a></li>
                        <li><a href="sms://+19011234567">Text ME</a></li>
                        <li><a href="mailto:jdc@big-server.com">Email ME</a></li>
                        <li><a href="facetime://109999">FaceTime ME</a></li>
                </ul>
        </div>
        <div data-role="footer">
                <h4>John Doe Investigations</h4>
        </div>
</div>
```

Joe Doe Investigations – Page 2

```
<div data-role="page" data-add-back-btn="true" data-theme="a" id="page2" data-title="JDC, Services">
        <div data-role="header">
                <p style=text-align:center><img src="JDC2.jpg" alt="JDC Logo">
        </div>
        <div data-role="content">
                <h2>My Services</H2>
                <h3><ul>
                        <li>Cheating Spouse</li>
                        <li>Disobedient Child</li>
                        <li>Disobedient Pet</li>
                        <li>Employee Absence</li>
                        <li>Stalker</li>
                        <li>Fraudulent Injury Claims</li>
                </h3></ul>
        </div>
        <div data-role="footer">
                <h4>John Doe Investigations</h4>
        </div>
</div>
```


JDC, Services
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Back
My Services
Cheating Spouse
Disobedient Child
Disobedient Pet
Employee Absence
Stalker
Fraudulent Injury Claims
John Doe Investigations

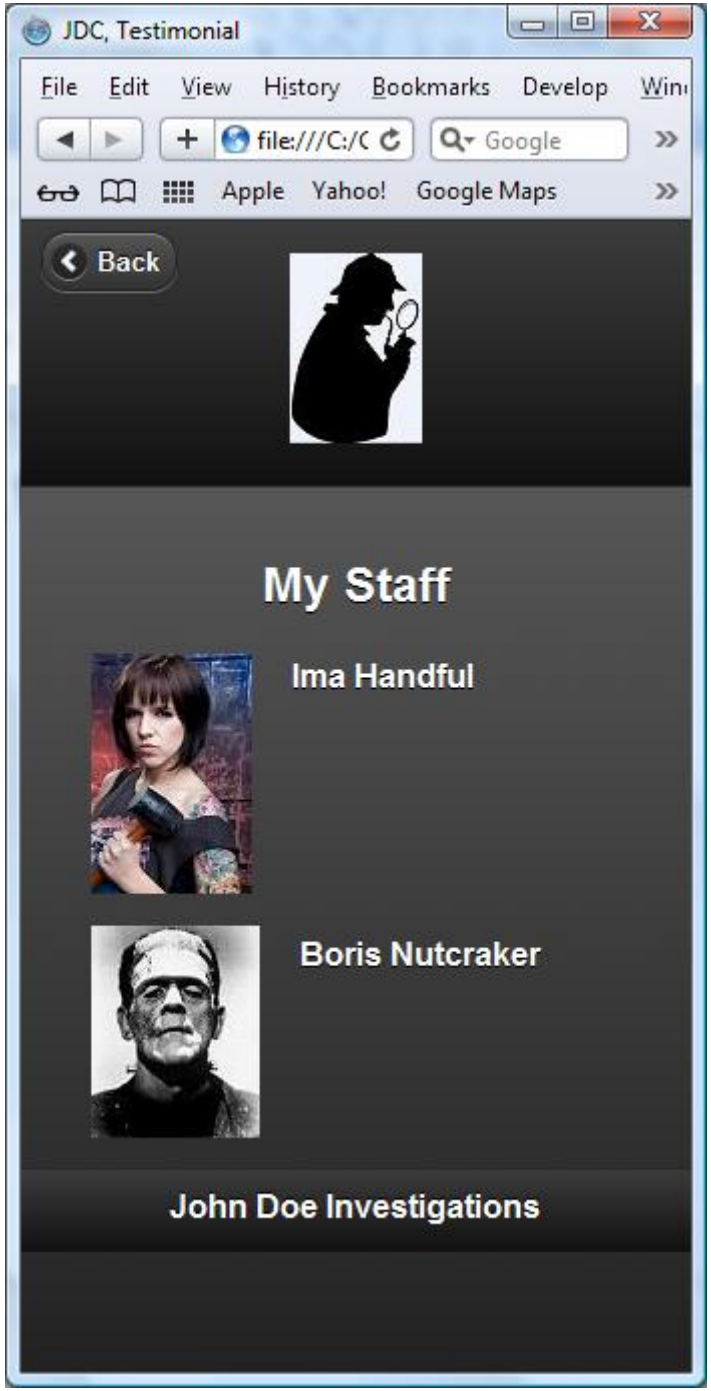
Joe Doe Investigations – Page 3

```
<div data-role="page" data-add-back-btn="true" data-theme="a" id="page3" data-title="JDC, Rates">
        <div data-role="header">
                <p style=text-align:center><img src="JDC2.jpg" alt="JDC Logo">
        </div>
        <div data-role="content">
                <h2>My Rates</H2>
                <h3><ul>
                        <li>Daily - $500/day</li>
                        <li>Hourly - $80/hour</li>
                        <li>Travel - $1/mile</li>
                        <li>Injury - Three times medical costs</li>
                </h3></ul>
        </div>
        <div data-role="footer">
                <h4>John Doe Investigations</h4>
        </div>
</div>
```


JDC, Rates
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Back
My Rates
Daily - $500/day
Hourly - $80/hour
Travel - $1/mile
Injury - Three times medical costs
John Doe Investigations

Joe Doe Investigations – Page 4

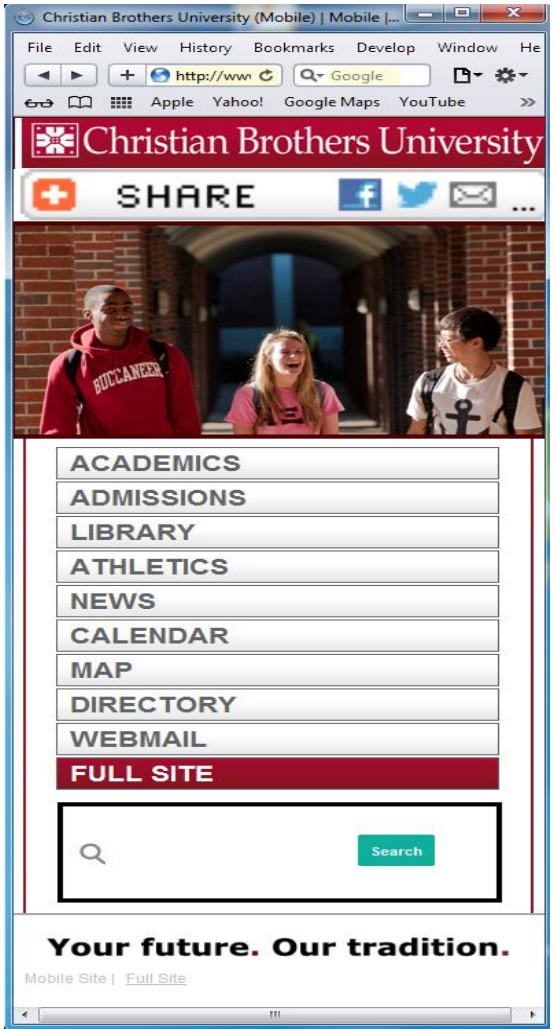
```
<div data-role="page" data-add-back-btn="true" data-theme="a" id="page4" data-title="JDC, Testimonial">
        <div data-role="header">
                <p style=text-align:center><img src="JDC2.jpg" alt="JDC Logo">
        </div>
        <div data-role="content">
                <h2 align="center">My Staff</H2>
                <img align="left" src="ima.jpg" alt="Ima" hspace=20>
                <h4>Ima Handful</h4>
                <br clear="all"><P><img align="left" src="boris.jpg" alt="Boris" hspace=20>
                <h4>Boris Nutcraker</h4>
        </div>
        <div data-role="footer">
                <h4>John Doe Investigations</h4>
        </div>
</div>
```

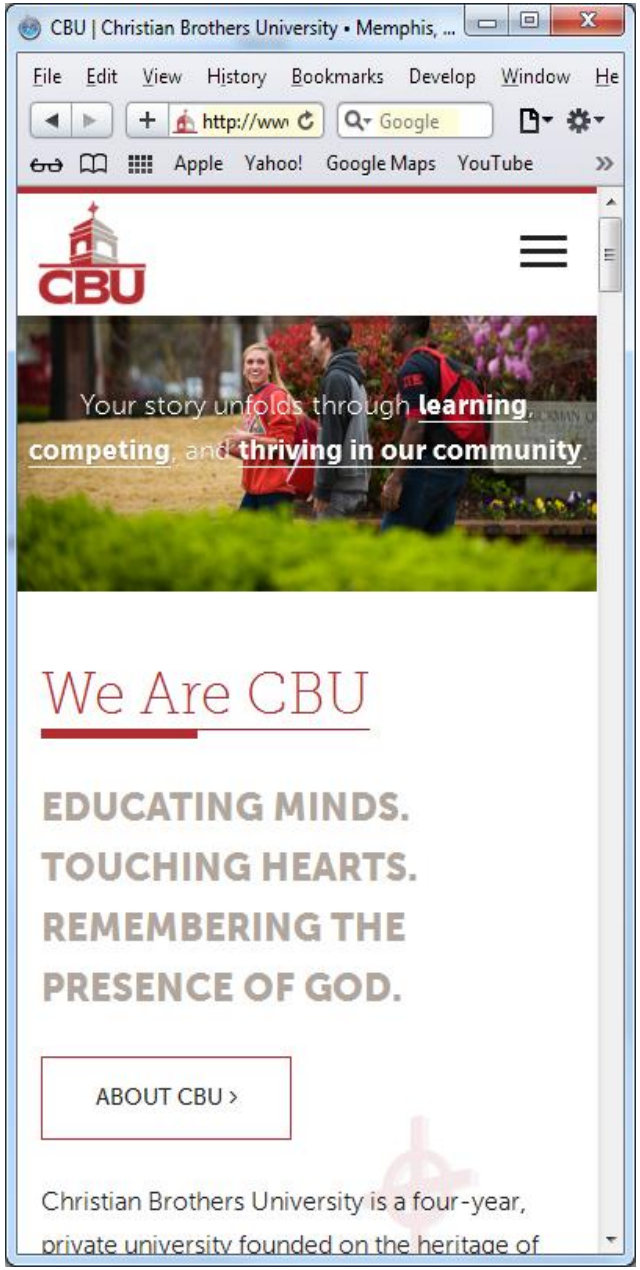

JDC, Testimonial
File Edit View History Bookmarks Develop Win
file:///C:/C
Google
Apple Yahoo! Google Maps
Back
My Staff
Ima Handful
Boris Nutcraker
John Doe Investigations

iCBU

Student Project Example

Current CBU Mobile Website

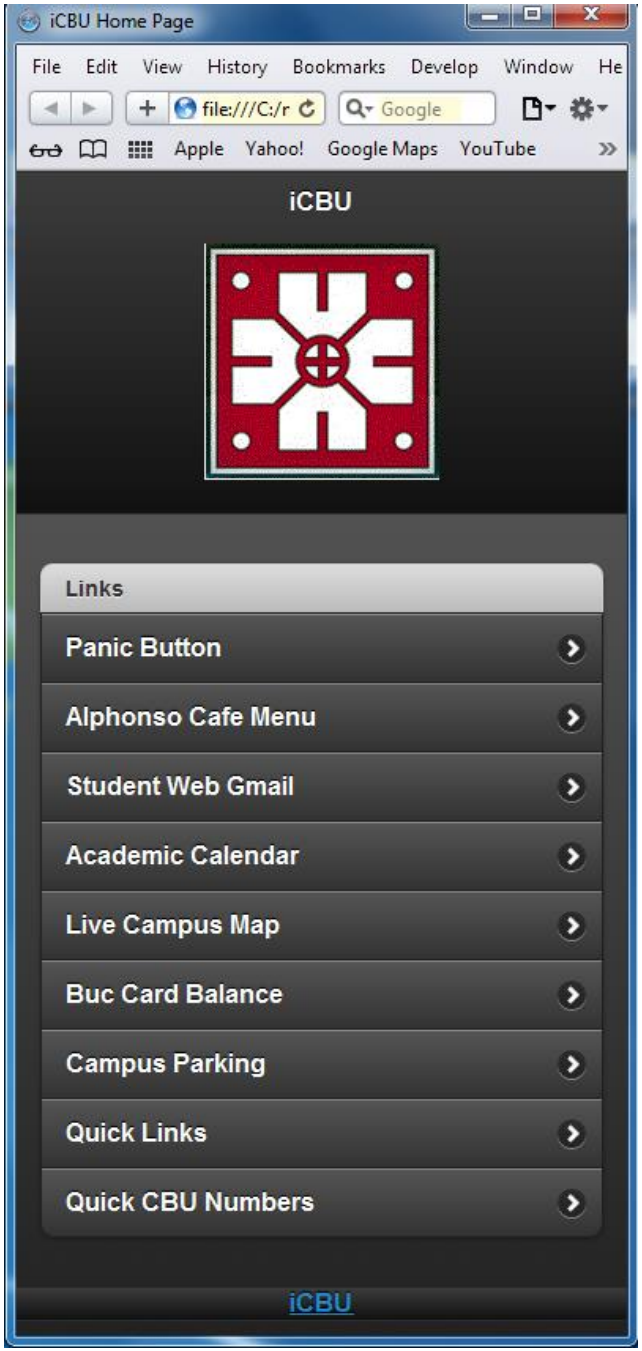




iCBU Project Goals

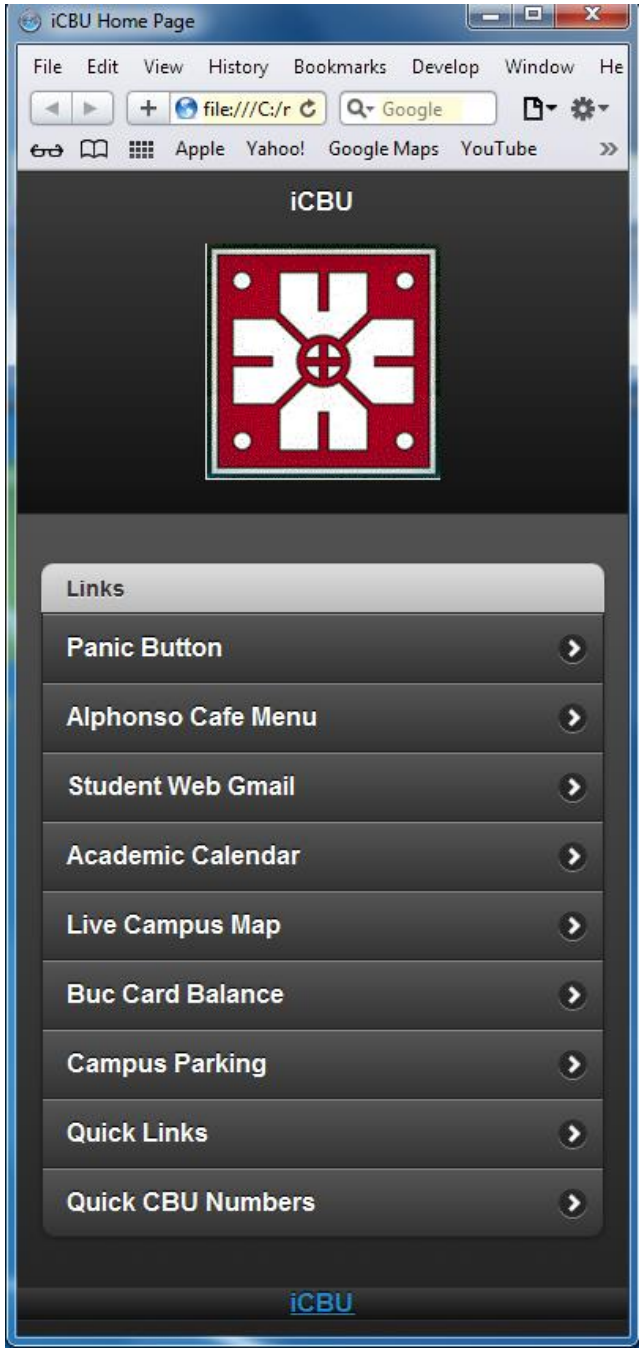
- Research other campus mobile web Apps
- Take advantage of unique features of smartphones (phone calls, text messages, GPS, camera, etc.)
- Emphasize the mobile capabilities that would be most desirable and most used from a student's perspective

iCBU Opening Page



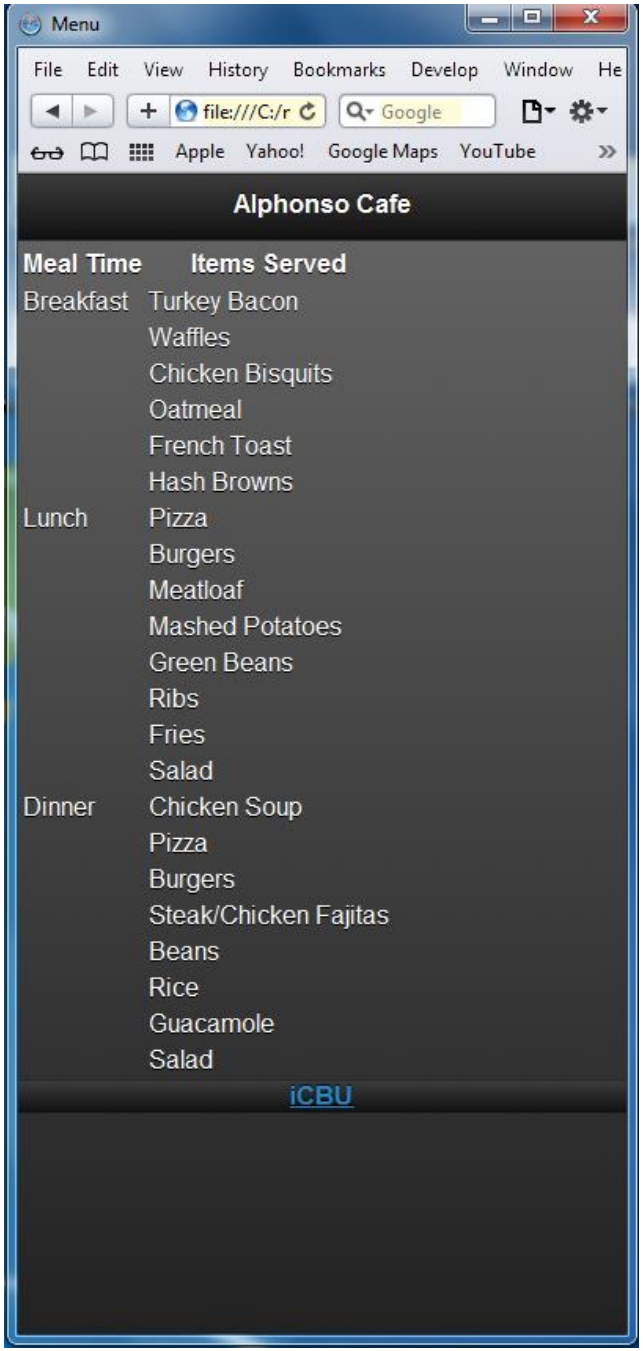
iCBU Panic Button

The “panic button” calls campus security and transmits the caller’s location (GPS if enabled or cell tower if not); then allows a voice call or text message.



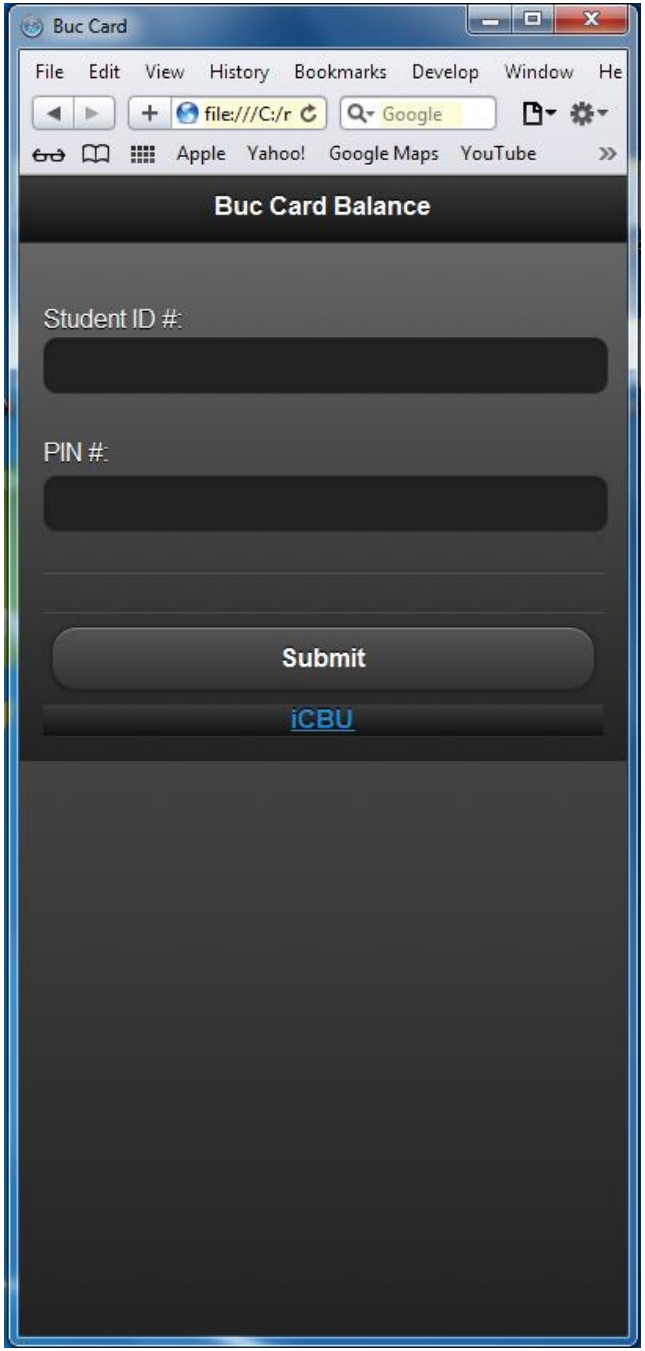
iCBU Café Menu

The “café menu button” shows what is being served for the current day (or next day if accessed after 8pm) in the main CBU cafeteria.



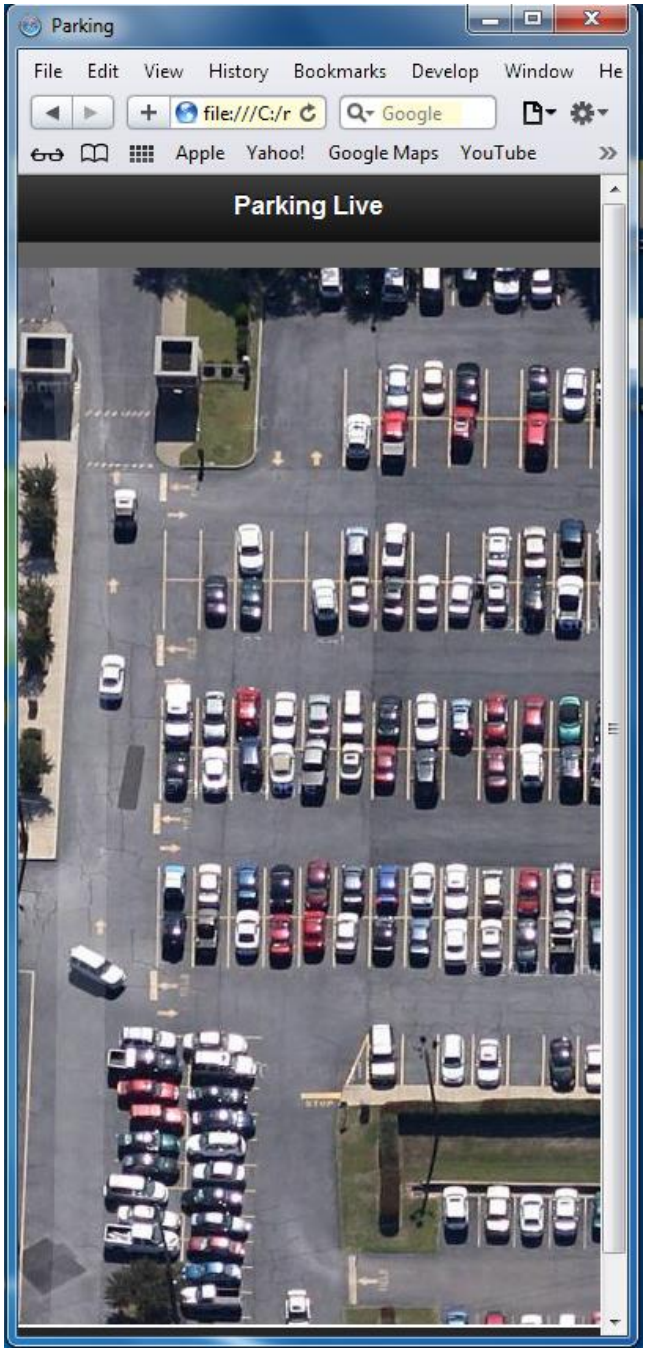
iCBU Buc Card Button

The “buc card” button allows the student to see the balance on their Buc Card (with the option to store the login info on the smartphone). Currently students do not have the ability to determine their Buc Card balance online.



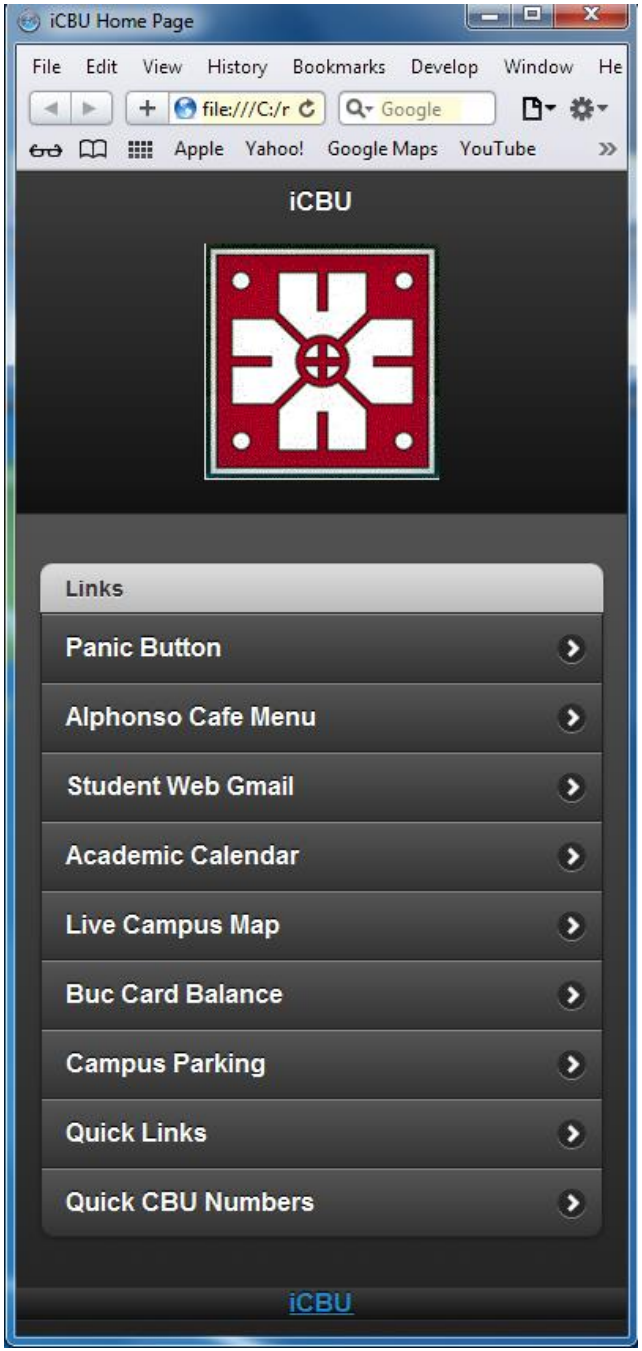
iCBU Parking Button

The “park button” allows students to see an up to date photo of the main student parking area. It would be possible via a camera mounted on top of the new science building with a “frame grab” each minute.



iCBU Mail Button

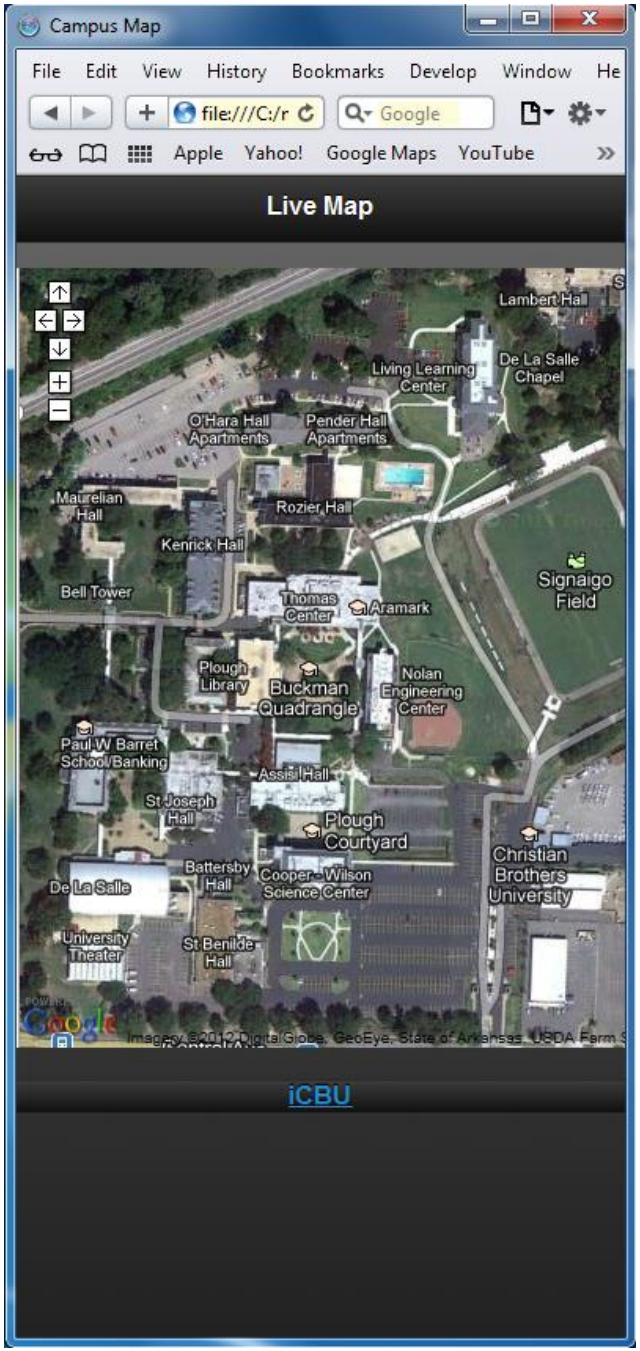
The gmail button goes directly to the students CBU web email account (with the option to store the logon info on the smartphone).



iCBU Campus Map

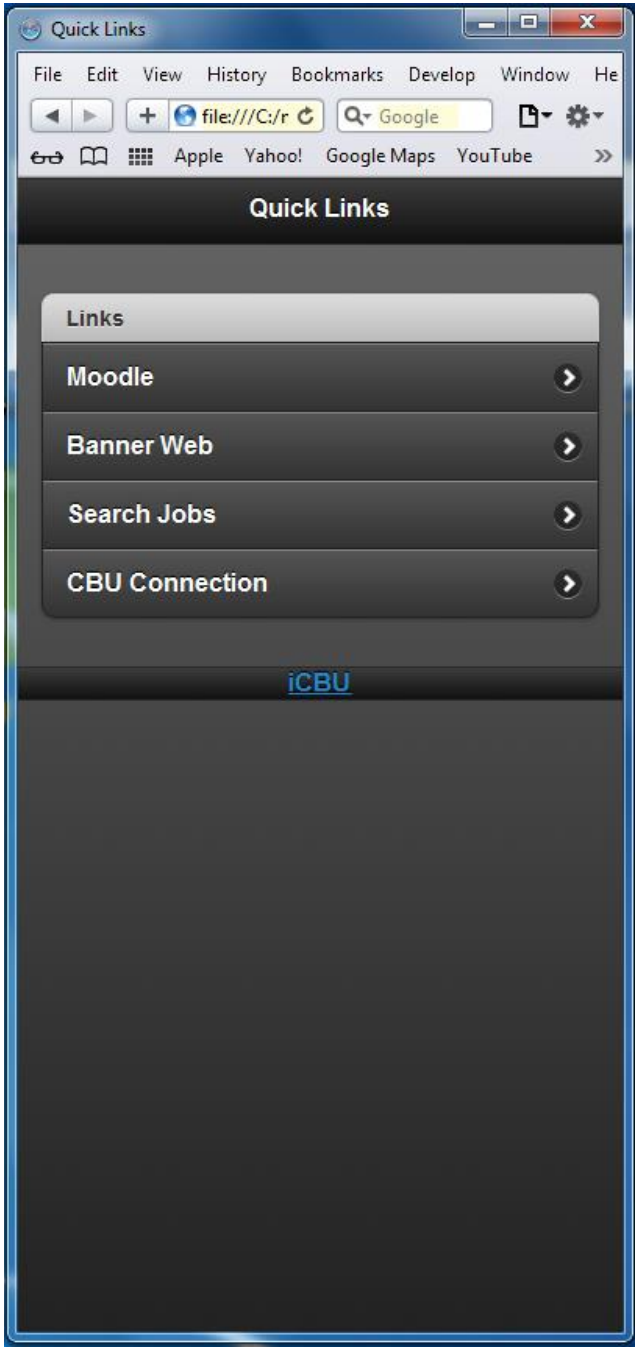
The campus map button opens an interactive map that allows navigation and zoom plus labeling.

If GPS is enabled, there is a “you are here” blinking star.



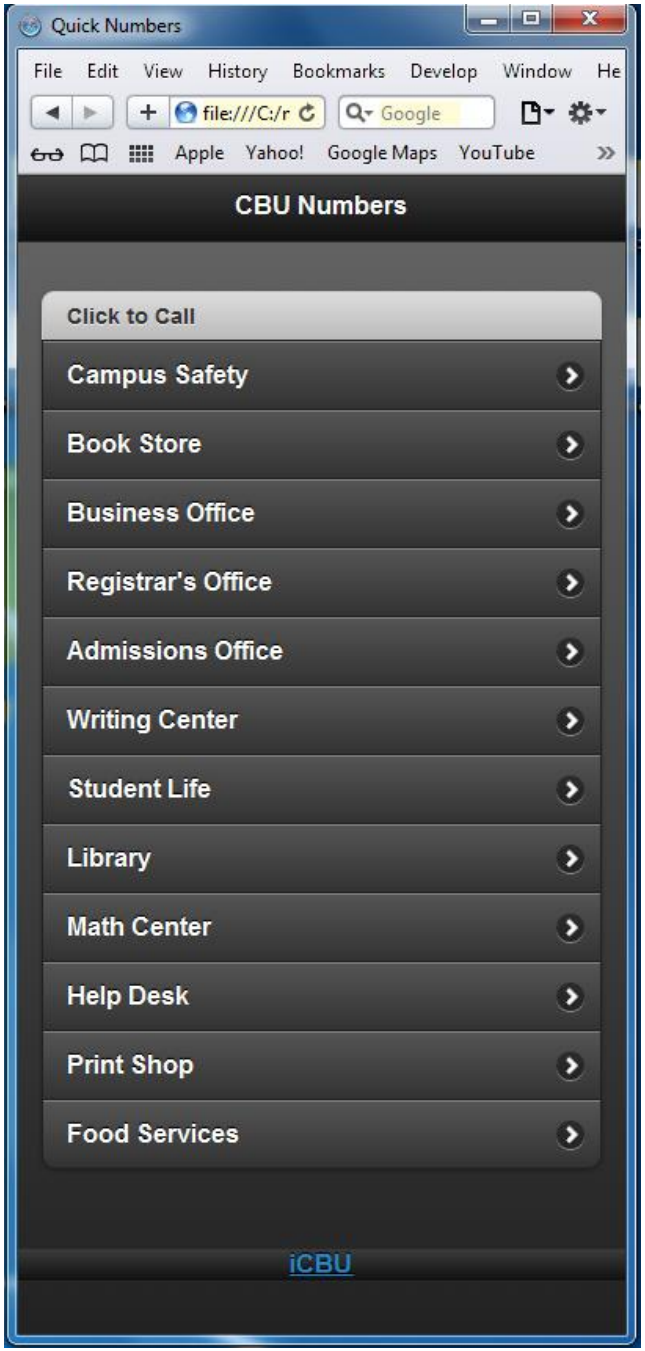
iCBU Quick Links

The “quick links button” would show the links that a student would most want to access from a mobile device. Some such links are shown here.



iCBU Phone Numbers

This directory page would allow a student to quickly find and auto dial key CBU phone numbers.



Other iCBU Main Buttons

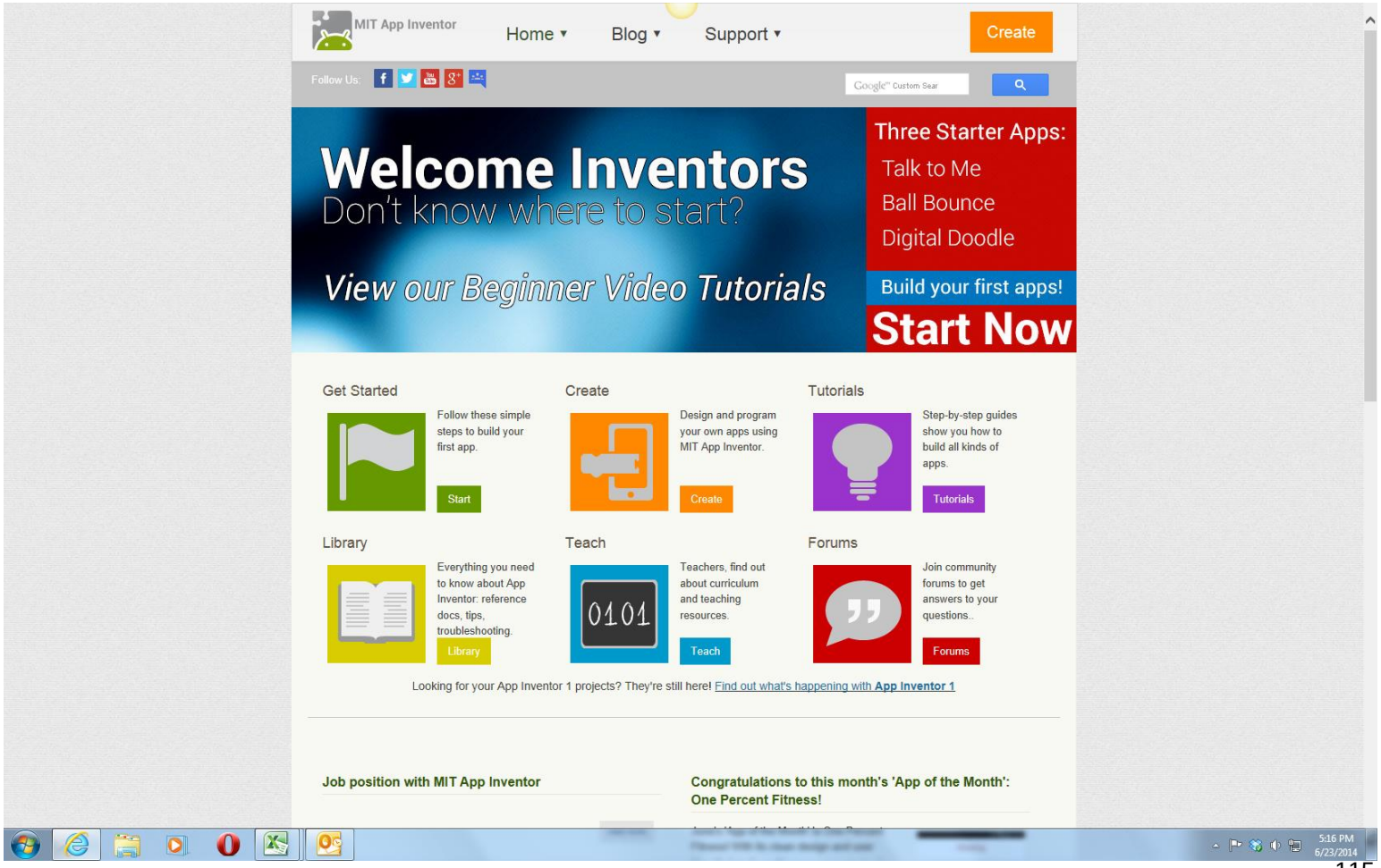
- Academic Calendar showing key dates: add, drop, withdrawal, etc.
- Campus Events
- Current gas prices at area service stations
- Sports schedules and results
- Area shopping “specials”

Mobile Development Frameworks

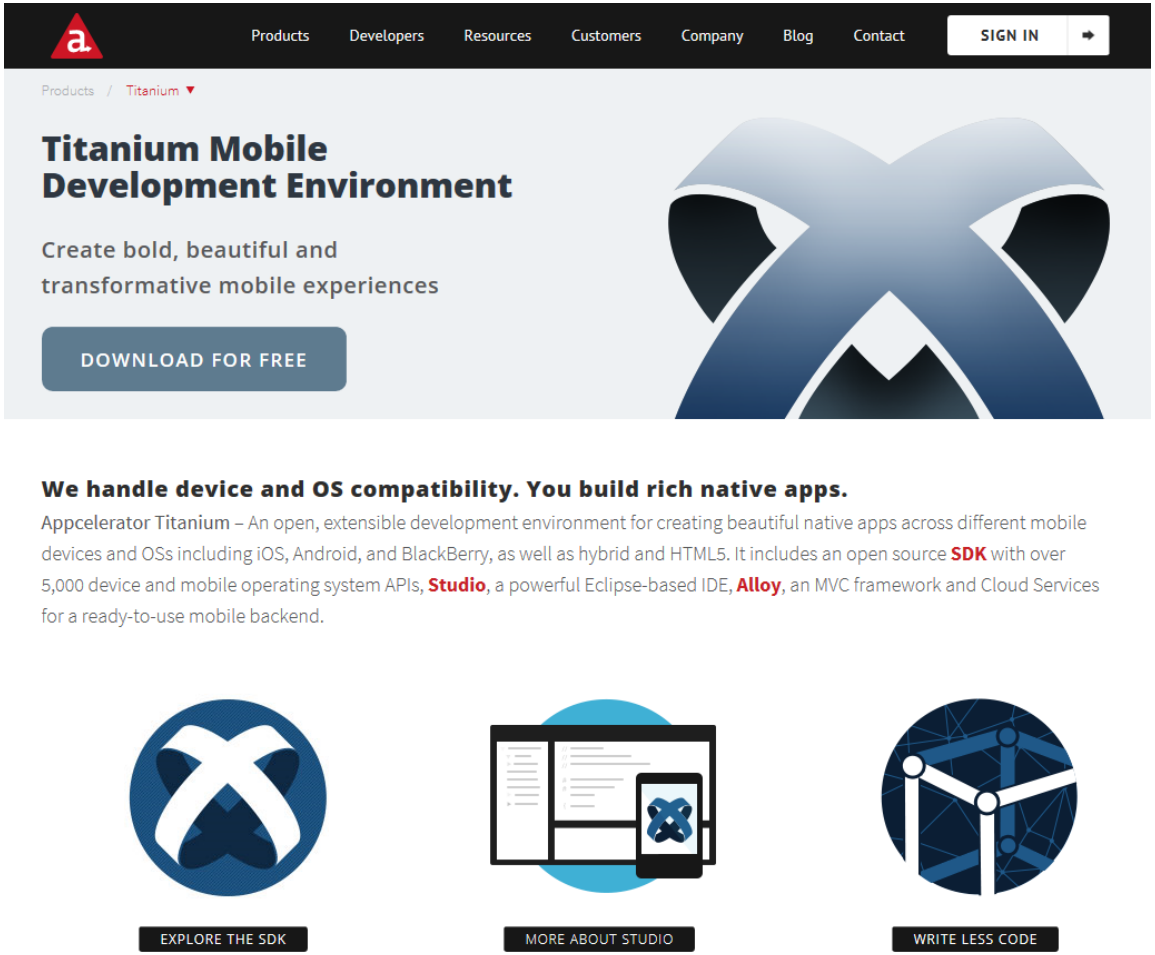
- There is a whole class of mobile development frameworks that allows you to code mobile apps using HTML, CSS, and JavaScript, as well as your favorite JS libraries (i.e. jQueryMobile)
- The frameworks then package your app for selected platforms so that it has access to device-level APIs that regular web applications would otherwise not:
  - Apache Cordova/PhoneGap
  - Appcelerator Titanium
  - Adobe AIR
  - Sencha Touch
- A lot of extra bells and whistles in addition to access to device-level APIs

App Inventor

[http://appinventor.mit.edu/explore/]



Appcelerator Titanium [http://www.appcelerator.com/titanium/]



PHP/Zend Based App System

Using Frameworks for Photos

- For example, most of these frameworks have excellent documentation on how to work with the smartphone camera without having to worry about individual device APIs

Photos via PhoneGap

- navigator.camera.getPicture(onSuccess, onFail, {
  - quality: 50, destinationType: Camera.DestinationType.DATA_URL
  - });
- function onSuccess(imageData) {
  - var image = document.getElementById('myImage');
  - image.src = "" + imageData;
- }
- function onFail(message) {
  - alert('Failed because: ' + message); }

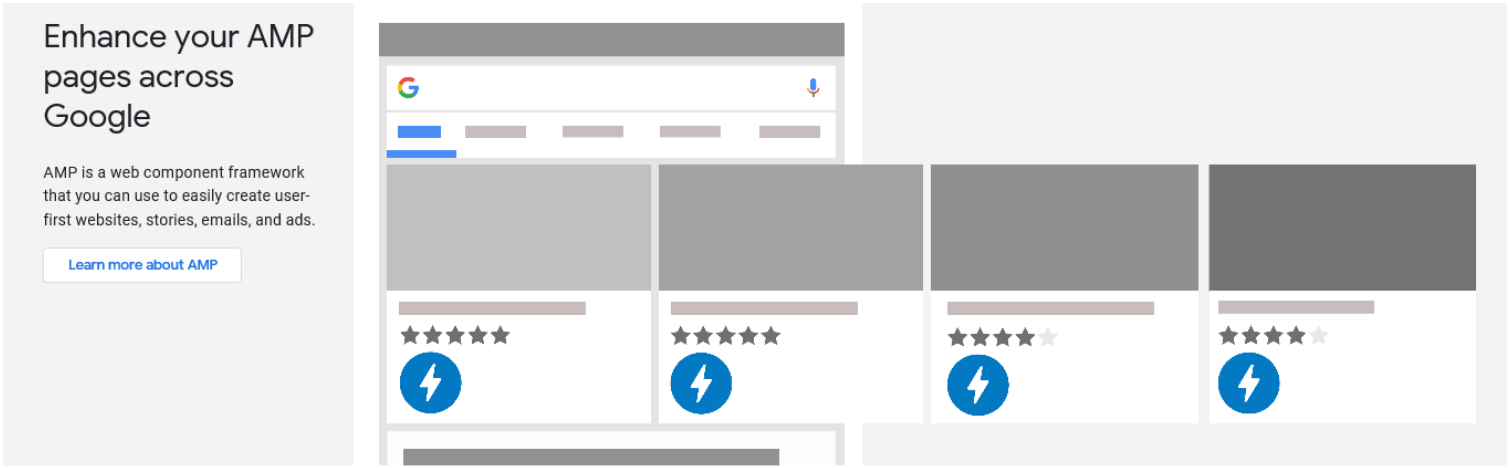
Photos via Sencha Touch

- Ext.device.Camera.capture({
  - success: function(image) {
    - imageView.setSrc(image); },
    - quality: 75,
    - width: 200,
    - height: 200,
    - destination: 'data‘
  - });

AMP

- The AMP Project is an open-source initiative aiming to make the web better for all by speeding up the downloading of static content
- AMP enables web experiences that are consistently fast, beautiful and high-performing across distribution platforms. AMP formerly stood for “Accelerated Mobile Pages”, but now works completely across desktop and mobile
- The official AMP plugin for WordPress supports fully integrated AMP publishing for WordPress sites, with robust capabilities and granular publisher controls.
- Features and capabilities provided by the plugin include:
  - **AMP-first Experiences**: enabling full-site AMP experiences without sacrificing the flexibility of the platform or the fidelity of content.
  - **Core Theme Support**: enabling AMP compatibility for all core themes, from Twenty Ten all the way through Twenty Twenty.
  - **Compatibility Tool**: when automatic conversion of markup to AMP is not possible, debug AMP validation errors with detailed information including the invalid markup and the specific components responsible on site (e.g theme, plugin, embed); validation errors are shown contextually with their respective blocks in the editor.
  - **CSS Tree Shaking**: automatically remove the majority of unused CSS to bring the total under AMP’s 75KB limit; when the total after tree shaking is still over this limit, prioritization is used so that the all-important theme stylesheet important is retained, leaving less important ones to be excluded (e.g. print styles).

AMP (con't)



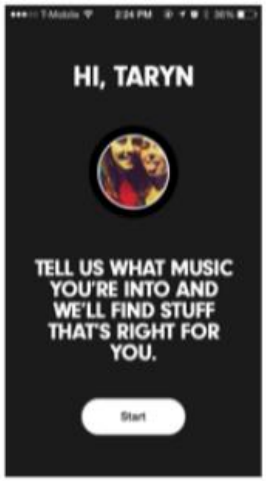
10 Common Mistakes to Avoid When Building Your First Mobile App

By Chris Preimesberger, eWeek

- Don't Bring Conventional 'Application Thinking' to Your App Project
- As Gartner Research has noted, "apps" and "applications" are not the same thing
- Applications are monsters prized for their long lists of capabilities, while apps are valued for doing a few things well—and for their purposefulness
- The temptation is to try to bring the do-it-all standard of applications to the do-a-few-things-really-well standard of an app

Best in Breed Mobile Sites

Best in Breed: Beats Music



Personalization Screen 1

Once user completes their personal information they are taken to the personalization phase. The messaging provided helps a user feel that this product has their best experience in mind.

Personalization Screen 2

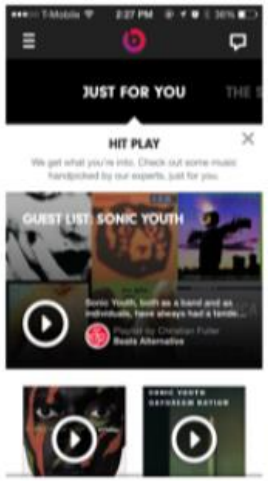
The personalization process provides directional guidance through a strong headlines and explanatory content to to guide the user through the process.

Personalization Screen 3

The provides intelligent progress information by changing simple language that lets the user know they only have to tap on 1 more artist they like, or they can continue adding and refining the collected data by adding more artists to the mix. The whole process is designed as if it were a game, engaging users to keep adding more data.

Personalization Screen 4

The progress bar that surrounds the user's profile photo let's them know that the personalization process is complete and they are given the option to finish or continue the personalization process.



Personalized Dashboard

With a user's personalized information is complete, they are given an immediate suggestion for a playlist on their main dashboard.

Best in Breed Mobile Sites

Best in Breed: Lulu Lemon

Personalization Screen 1

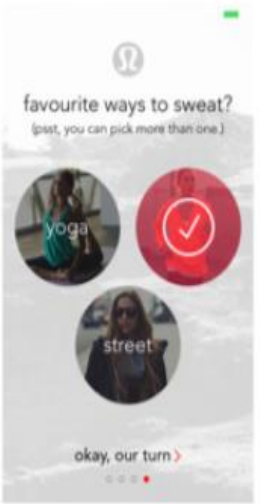
Lulu Lemon's on-boarding process immediately lets a user know that they are collecting information to personalize their results.

Personalization Screen 2

On-boarding process screens utilize the opportunity to present branding look and feel while gaining useful information at the same time.

Personalization Screen 3

Visual elements and selection language keeps the process simple using strong CTAs and selected states.



Personalization Screen 4

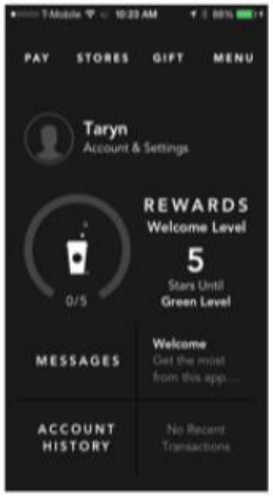
Messaging is used as another way Lulu Lemon builds brand identity and engages with the user in a fun and helpful fashion.

Personalized Product Listing

A personalized product listing screen which shows related products based on user's selected preferred workouts during the personalized on boarding process.

Best in Breed Mobile Sites

Best in Breed: Starbucks



Dashboard

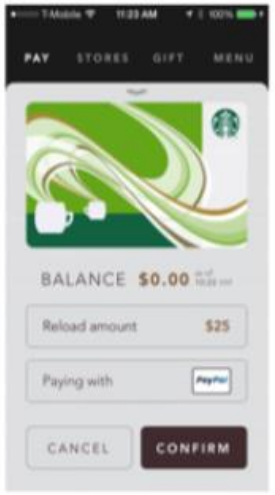
The dashboard showcases a simplistic navigation system where each tab at the top initiates a scrollable sliding modal. This action teaches the user that they can access the dashboard from any of these primary navigation points.

Starbucks Card Overview

From the Pay menu, a user is offered useful gesture-based interactions such as adding a new card (swipe right), editing their cards (swipe up), or selecting a card to use for payment (tap).

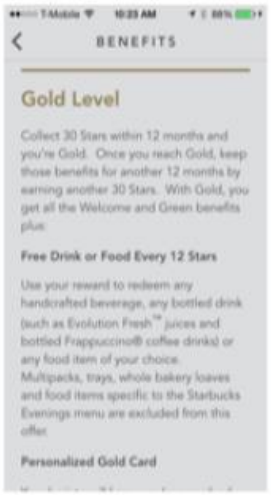
Payment Scan Screen

A user can tap on the card they want to use which flips to show a scannable bar code.



Reload Payment Screen

Adding value to a user's card is fast and easy with the option to pay with card or use a Paypal account.



Benefits Info Screen

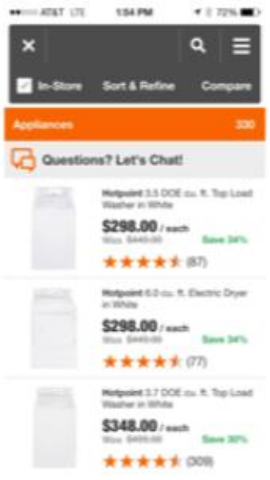
This screen shows the different reward levels and what benefits the user can receive.

Best in Breed Mobile Sites

Best in Breed: Home Depot

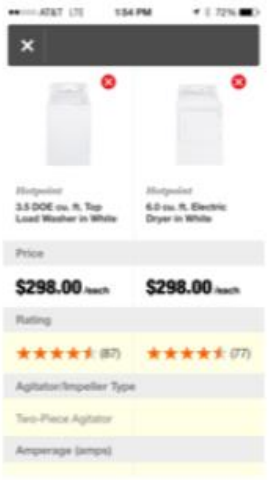
Home Screen

The home screen groups several actions in the top navigation, including search, barcode scan, your store, and the side menu. They are able to compact all of these items into a small space, which gives the rest of the screen real estate to showcase products and promotions.



In-Store Deals Screen

In-Store Deals are easily accessible from the home screen and users can filter by products that are in-store.



Comparison Screen

Users are able to compare products and view features side-by-side. This is very useful in helping customers come to a decision to putchase an item without leaving the app.

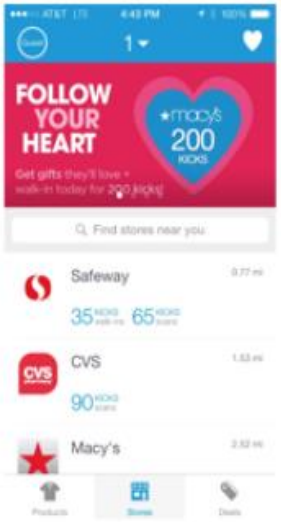
In-Store Guide Screen

This is a great tool for users who are trying to locate items in-store. The map is 3D and users can zoom in, pan, and rotate the map. Additionally, users can find individual items by going to the product SKU screen and tapping the "In-Stock" badge.

In-App Chat

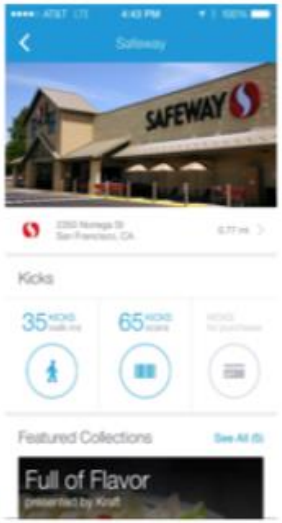
If users need help they can chat with live agents within the app. The feature is not available all the time, but it is extremely helpful and offers the ability to send photos to the agent.

Best in Breed Mobile Sites



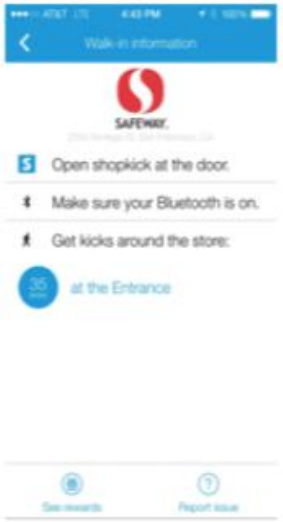
Home Screen

Shop kick does a great job of driving customers to brick and mortar stores by incentivizing them with offers and points.



Local Stores Screen

By personalizing each brick and mortar store with metrics like walk-ins and product scans, users feel a stronger connection to that store and are more compelled to visit more often. This also adds some gamification where users are rewarded for frequent visits and purchases.



In-Store Offers

Presenting highly relevant and contextual in-store offers is a great way to engage customers during the shopping experience. Offers can include exclusive discounts or iBeacon notifications alerting them to new products.

Scan Deals Screen

Barcode scanning allows customers to interact with products and unlock exclusive in-store deals and offers. This is a great way to get customers to learn about or drive traffic to a certain product.

Deals Overview Screen

The deals overview screen is a great way for users to quickly find the latest and greatest deals and offers.

References

- Murach's JavaScript and jQuery by Zak Ruvalcaba , Mary Delamater , et al.
- jQuery Mobile Web Development Essentials by Raymond Camden and Andy Matthews
- HTML5 Mobile Websites: Turbocharging HTML5 with jQuery Mobile, Sencha Touch, and Other Frameworks by Matthew David

Homework

- Textbook Chapter 5
- Optional, extra credit:
  - Modify your project 3 web page to work well on both desktops and smartphones
  - Or make a second smartphone version of your project website
- Appendices →

Appendix

MOBILE DEVELOPMENT GUIDELINES

Mobile Development Guidelines

- With the advent of mobile devices, a new industry came into existence. Mobile devices are now so popular that many users no longer buy desktop or laptop computers. Advertisers, seeing the value of this new medium are taking for advantage of it, offering products, games, apps and more. In this article you'll learn about 10 design practices for building mobile apps. These practices will help you get the results you seek and also satisfy your customers.

- **Before You Begin, Consider Your Audience:** Before you take any time to build an app, consider your audience. What do you hope to achieve? How do you envision your audience using your app? These are important questions to consider up-front.

- **Check the App Stores:** Many times people come up with a great idea for an app and start to brainstorm how to build it. There's only one problem. Despite how unique you might think your idea is, there's an excellent chance that someone might have already built it, or something similar to it. If that's the case, you would be wasting a ton of time (and money). If an app already exists, you can use it as a template to create your own product, or you might consider partnering with the creator(s) of that app and using it as part of your strategy.

- **Involve Potential Users in the Design Process:** One danger of any design process is working only with your team and not involving the end users at all. Then, when the design is done and is released to the public, some or many aspects of your design might not translate well to the real world. To avoid this problem, involve potential end users in the design process and use their feedback to make changes as necessary.

- **Create a Storyboard:** The storyboard is one of the most important aspects of the design process. This is where you lay out the complete functionality of your app on paper. If there are problems, you can resolve them at this stage. The storyboard allows you to plan out all aspects of the design, including future components, such as plug-ins.

- **Make the App Easy to Understand:** The app should be easy to understand with descriptions to accompany graphics (if necessary) and additional instructions. One design flaw is relying too much on images to tell the tale. That's a major error because users might not be able to figure out the purpose of your app if you use a lot of graphics. Clear instructions are necessary.

Mobile Development Guidelines (con't)

- **Avoid Overuse of Graphics and Animations:** Both graphics and animations can add a nice "Wow" factor to your app but there's a major downside – slow loading times which translate into a poor user experience. Whenever possible, either avoid the use of bitmaps or animations or limit their use to only essential features. And if you do use graphics, use vector graphics whenever possible. The files sizes from these are much smaller, so they'll load faster.
- **Consider the Sizes of Buttons and Icons:** When working with a mobile interface, you have a limited amount of space and some designers add too many buttons/icons. Another consideration is the size of the human finger tip. If the buttons/icons are too small, users could make errors with selecting the wrong one. Likewise, if there's not enough space between the buttons/icons, that can cause trouble as well. If in doubt, test your layouts and get feedback.
- **Create a Core Application:** This means taking the most important features and building those into a core application experience. Additional functionality can be created by building plug-ins that can be purchased as necessary by the user. This avoids overloading the core part of the app with too many features.
- **Create a Consistent Workflow:** This translates into making sure the user experience remains the same on all platforms. If you change that for each device, you'll confuse and annoy your users.
- **Test the Design:** With any design, this is the most important aspect. If you've been following the strategies listed in this article you'll be testing your app every step of the way. Still, it's important to test the finished product and not only once but several times with different users. If there are problems, fix them, then test the result again.
- **Conclusion**
- As you can see, creating a mobile app requires a lot of strategic thinking before you even begin to build it. While it's important to consider the design, it's more important to search the various app stores first, because someone may have built an app that's very similar to what you have in mind. If that's the case, finding a way to work with the existing developers could be of great benefit, and not only to save on development costs. If a business relationship isn't possible, you still have access to an app that will offer you a template for your own design.

7 Things You Must Avoid When Designing Mobile Apps

- **Coding Without a Plan:** If you're building an app from scratch, this is the most important step. If you code without a plan you could end up with a real mess. Not only will the process take longer, it's likely to be haphazard and expensive. To save yourself the grief, use a storyboard or some sort of flowchart system where you build the entire application on paper, complete with diagrams if necessary. If problems are found, they can be addressed and resolved. And once you've built the storyboard, you've built the app. All you need to do now is to write the code.
- **Not Considering the Screen Size:** This is critical. The screen size varies with each mobile device so it's important to take that into consideration when creating your layouts and addressing scaling and display issues as they arise. When building an app, it's very tempting to want to put a lot into the interface. That's a bad idea for several reasons: You could overload the user with too many buttons, the interface could be confusing, or it might take too long to load. Instead of trying to put as much into the app as you can, it would be better to design a core system with the opportunity to add plug-ins later, if the user desires. This decreases the overhead, it will make the design process easier, it will allow for scalability and ongoing profit streams after the fact.
- **Not Using Templates:** Whenever possible make use of templates and code snippets when building your app. These will help you save time and reduce your costs. Some programs, such as Adobe Dreamweaver and Topstyle 5, make use of templates.
- **Using Text in the Icons:** Don't use text in the icons. This is a mistake because both Google Play and Apple use text next to the icon, making it totally unnecessary. Instead, concentrate on the design of the icons. Make sure the icons use smooth transitions and well-defined edges. You want them to stand out from the background, not blend into it.
- **Not Informing the User When Programs are Loading:** With many an app, sections of it might take time to load. In some cases, developers don't let the user know that this is happening. If that's the case, it could create the following consequences: The user might be patient and wait for a few moments, but likely not. Chances are, the user might think the app stopped functioning and restart it, or, if there's still no response, quit the app and in an extreme case, uninstall it completely. This is why it's so important to let the user know when sections of an app are loading.
- **Not Paying Attention to the Different Operating Systems:** When building apps, you'll be working with different operating systems and you'll have to design your apps accordingly. With that in mind, the interface used for an iphone will be different from the layout for an Android. It's really important to make sure that your interface design matches the layout of the OS. If not, you run the risk of confusing and/or annoying the users, which will directly affect the adoption of your app. In combination with this step, it's important that you take color into consideration and make sure it fits the environment.
- **Not Testing the Interface before Deploying:** This is one of the most common issues, not testing before deploying, yet it's one of the most critical of steps. If you miss an important part of a procedure, the entire process could fail. When you test, make sure you test your app with people who have never set eyes on the product. That's the true acid test. It's here that you'll find out whether you built the app in a way your target audience can use it, or if you missed/omitted steps. Once you get the feedback, you'll know whether all is well or whether some redesign work is necessary.
- **Conclusion**
- When beginning any design process, it's important to think about the end result, first. To ensure the success of your designs it's important to involve users in the design process. Nothing is worse than working in a vacuum. What you and your design team might think is fabulous could be a complete flop with the users, so test early and test often.

Programming and Documentation

- There isn´t a lot of difference between code such as HTML, PHP, or C; code is code
- A lot of the languages overlap and are actually very similar to each other in how they function
- Good code is, number one, documented so someone else could read it and know what it is doing
- The documentation is often in the form of a WIKI
- Good structure means that an analyst has asked the client what they want, what they want to achieve, then a business case is built around it
- After that the business case is broken down into its individual components, which then get assigned to programmers for fulfillment
- The reason that most code is problematic, is that most documentation is horrible is because no one likes documenting and they wait until the last minute to do so
- If you're a programmer, all you want to do is program then move onto the next thing
- And if you're a one-man programming shop you don't need to explain to anyone else
- When you're in an organization of 3, 5, or 10 people, inevitably some people will ask why you programmed a certain way and the reality is that logic doesn't flow the same way for everybody
- If you ask people how to program something, you will get 10 different ways to get it done
- Elegant code is small; the fewer lines of code you can accomplish something in, the better

Programming …

- The next thing to remember is you should name your variables so you know what they are, later
- And if you use layers you should label what each layer is for
- Good code should call the server as little as possible
- The more times you have to call the server the slower it goes, the slower your SEO rank, and the lower the experience for the customers
- When using CSS it is better to have one main sheet and you reference chunks of it, rather than one style sheet per page
- With a website that has thousands of pages that can be a bit crazy
- Again, every time you have a style sheet, you have to hit the server and the server has to go back and forth
- If it's the same style sheet for everybody you've already downloaded it when you've hit the first page so if you hit the 5th, 20th, or 30th page within that browsing session, it doesn't have to download a different style sheet

Variable Type Limits (overflows)

- When building an HTML5 app it's really important to consider the upper limit and to design your variables accordingly
- When Youtube was built, they had an upper limit on the number of views they could show, which topped out at around 2 billion; the video which finally "broke" Youtube was: "Gangnam Style (once Gangnam Style surpassed that number, every view past that didn't count)
- When you create variables, these can change from one value to another. When you create a variable you set what that variable is going to be. If the variable is an integer it's always going to be a number, but an integer variable only holds so much space (such as 65535)
- That is the limit of an integer on certain platforms. If I had a value of 65536 or higher, that variable explodes. This results in errors, crashing, security leaks, etc. A long integer (long int) is the same as an integer, just bigger
- With the Youtube video, people could still see it but the counter stopped working. It's exactly the same as the Y2K bug
- "Youtube never thought that anyone would get 2 billion views. So they had to swap out that integer to the next largest number
- This is an example of HTML5 coding where it makes a difference as to where you put things and how you place them. Back in the day when they were building Youtube they were thinking about most ridiculous number of views that anything has ever gotten and they came up with 2 billion which seemed crazy. And they thought that if they set the value to that, no one would ever hit it."
- "Youtube was founded on February 14, 2005 and from an Internet perspective, it's ancient; that's why one should design the workflow first, so you don't fail when it comes time to execute and to deliver to the customer what they want

Session Management

- When programming a project, the first thing to do is figure out what each screen will look like, then figure out how each screen flows to the next, and what data must be kept from one screen to another
- As an example, the login data has to be kept the entire time the session is running if you want to access your account; it's resource management, and HTML5 is still HTML, therefore there are a limited number of sessions that you're able to utilize while in the browser
- A session is an environment where you bring something across from one page to another
- HTML is stateless, it has no idea whether you're connected or not, and a login is really important
- That session keeps your login from page to page; as an example is if I want to use HTML5 and log into my bank account - if I don't keep that session from one page to the next I'm going to have to log in every time the screen changes
- To make it work you would use a security token and it keeps you logged in from one page to the next and you want that session to be open for the entire time that you're logged in; that's also why those sessions have timeouts (if you have ever logged into your account and you leave it for a bit and come back you'll see a message that your session has timed out)
- The session allows you to log-in, follows you around and then counts down from, say, 20 minutes and if you're no longer there after 20 minutes, it kicks you out
- If I access my bank account from the library I expect it to kill the session after I log out and if I forget, to kill the session after a specified amount of time
  - With a session, you have a limited amount of them and you don't want to have too many open pages If you do you have a lot of memory moving back and forth. If you've been working on a web page and it randomly crashes, this is because you have too many things open. If your pages become sluggish, one way to fix the problem is to close the browser and open it back up again.

Native Device Code

- There are some important differences between HTML5 and native code and when you would want to use one or the other or both
- Native code allows you to touch all the resources of the phone, whereas HTML5 does not
  - As an example, if I want to access the camera, one can't on HTML5 but one can on the native app
  - Any time you want to use any of the peripherals of the phone HTML5 may or may not have reach to it but the native code always has reach to it
  - For Android you would use Java and if you have an Android app
  - If you're working on the iPhone, that would be Xcode or Cocoa
- The main benefit to HTML5 is if I make it once, it works on everything, in contrast, native code will only work on the platform you built it on
- It's also important to realize there will be some instances where you will want to use both HTML5 and native code to get the result you want.
  - Let's pretend we have an app that's a game and we want to sell a sword
  - We have to use the Apple or Google store to sell that within the app and make it happen
  - Now, what does it cost one to give some guy a digital sword?; Nothing
  - Let's juxtapose that with an app that runs a café and I want to buy a sandwich
  - There's a cost for that sandwich and we have to manufacture those
  - Apple and Google take 30% of whatever you sell through the app or Google Play store
  - This is where you can use HTML5; if I have a pre-existing web store and it is not a digital product, my HTML5 shopping cart will solve the problem and allow me to save that 30%

Apps Should Be Purposeful, Not All-Purpose

- As a rough guide, make a list of desired features for your app, then delete half of them
- For every feature you introduce into your backlog, one must come out
- Finally, embrace the clarifying "Darwinism" of app stores, which reward simple, purposeful mobile experiences while ruthlessly eliminating those with feature bloat

Do Not Fail to Build for Multiple Platforms

- Failing to build for multiple platforms is no less than a decision to ignore entire user segments—justifiable in certain cases, but probably not something to make a habit of
- Given the cross-platform app development tools that exist, there's no reason not to build apps optimized for a multi-OS, multi-device world

No Apps Without APIs

- Good mobile apps are greedy things, hungry for all manner of data from enterprise systems to SaaS repositories, public sources such as social and the looming Internet of things
- This is where application programming interfaces (APIs) come in
- They give developers the simplified access to the data and services needed to build amazing apps
- In fact, good mobile APIs act as a spur to innovation
- Think of them as Lego blocks: The better and more varied the collection of blocks you make available, the better and more creative the objects people build

Agile Isn't Fast Enough

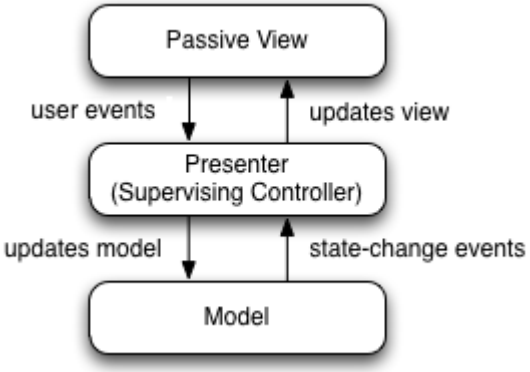
- Building mobile apps well means optimizing your whole delivery process around velocity
- Users expect a steady stream of feature updates, and each new release of the operating systems will demand app updates even if your users don't
- The only way is to embrace an MVP mindset—that is, "minimum viable product“
- This is a strategy of putting the bare bones of necessary functionality in front of users as quickly as possible
- Why? To get something in their hands fast and to improve the app based on their actual use
- It's all about maximizing learning and minimizing resource spend on the wrong things

MVP Approach Isn't Easy, but Don't Ignore It

- The MVP approach is not easy; it depends on employing analytics to find out how users are actually working with an app
- It requires discipline and a willingness to listen
- The key, covered in the next slide, is real-time analytics that show you in an instant how users are (or are not) using your app

MVP (con't)

- **Model–view–presenter** (**MVP**) is where the model–view–controller (MVC) architectural pattern derives from, which is used mostly for building user interfaces
- In MVP the *presenter* assumes the functionality of the "middle-man"
- In MVP, all presentation logic is pushed to the presenter

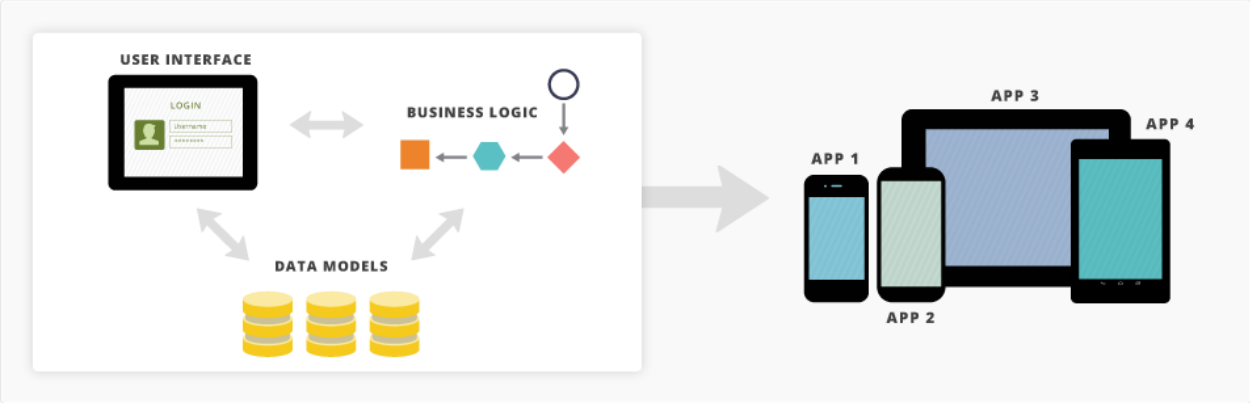


MVP (con't)

- MVP is a user interface architectural pattern engineered to facilitate automated unit testing and improve the separation of concerns in presentation logic:
  - The *model* is an interface defining the data to be displayed or otherwise acted upon in the user interface
  - The *view* is a passive interface that displays data (the model) and routes user commands (events) to the presenter to act upon that data
  - The *presenter* acts upon the model and the view. It retrieves data from repositories (the model), and formats it for display in the view

MVC/MVP in Acclerator

Architecture



Using Appcelerator Alloy, developers will realize the benefits of investing less time and cost and writing even less code.

Features

- Standard MVC framework built on Node.js with Backbone.js and Underscore.js support
- Fully code your app using XML, CSS and JavaScript
- Use Alloy CSS themes to manage your app's look and feel
- Easily bind data sources to your apps' user interface
- Reusable widgets support
- Offline storage support
- Fully integrated with Studio

You Can't Manage What You Can't Measure

- So you've got your app out there in the wild and people seem to like it, but do you really know how well it's doing?
- We're increasingly moving into a mobile world, and with mobile comes a wealth of information unlike anything we've ever seen before
- This includes variables like user location, device type, app version, operating system and device orientation
- The trick is capturing this data, making quick sense of it and using it to inform your next release
- Without analytics, you're flying blind and your mobile app plan will amount to little more than dart-throwing
- The solution: Implement analytics with your first app, sort through the findings and improve continually with each subsequent release

You're Not Smarter Than Your Users Today

- Users expect anytime access to elegant, easy-to-use services, all running on their device of choice
- Too often businesses forget the demands of the end user when creating their first mobile app
- Start building your app with the end user in mind and optimize around "mobile moments“
- As Forrester Research calls them, to create targeted, context-aware, anytime/anywhere experiences that people love
- And don't stop once you've got your app up and running; apps require ongoing care and feeding and dedicated resources
- Think of your apps as products, not projects

Don't Just Talk About Security

- With countless devices working on multiple operating systems, many levels of risk and vulnerabilities exist, increasing the complexity and importance of securing mobile devices
- Appcelerator advocates six layers of mobile app security, depending on the nature of the app:
  - 1) authentication and authorization of users
  - 2) encryption for data in motion
  - 3) encryption for data at rest (client-side)
  - 4) encryption for data at rest (server-side)
  - 5) app code security via native source file encryption
  - 6) security for app distribution and management
- Obviously not every layer applies to every app type, but failure to consider each layer can lead to some unhappy headlines

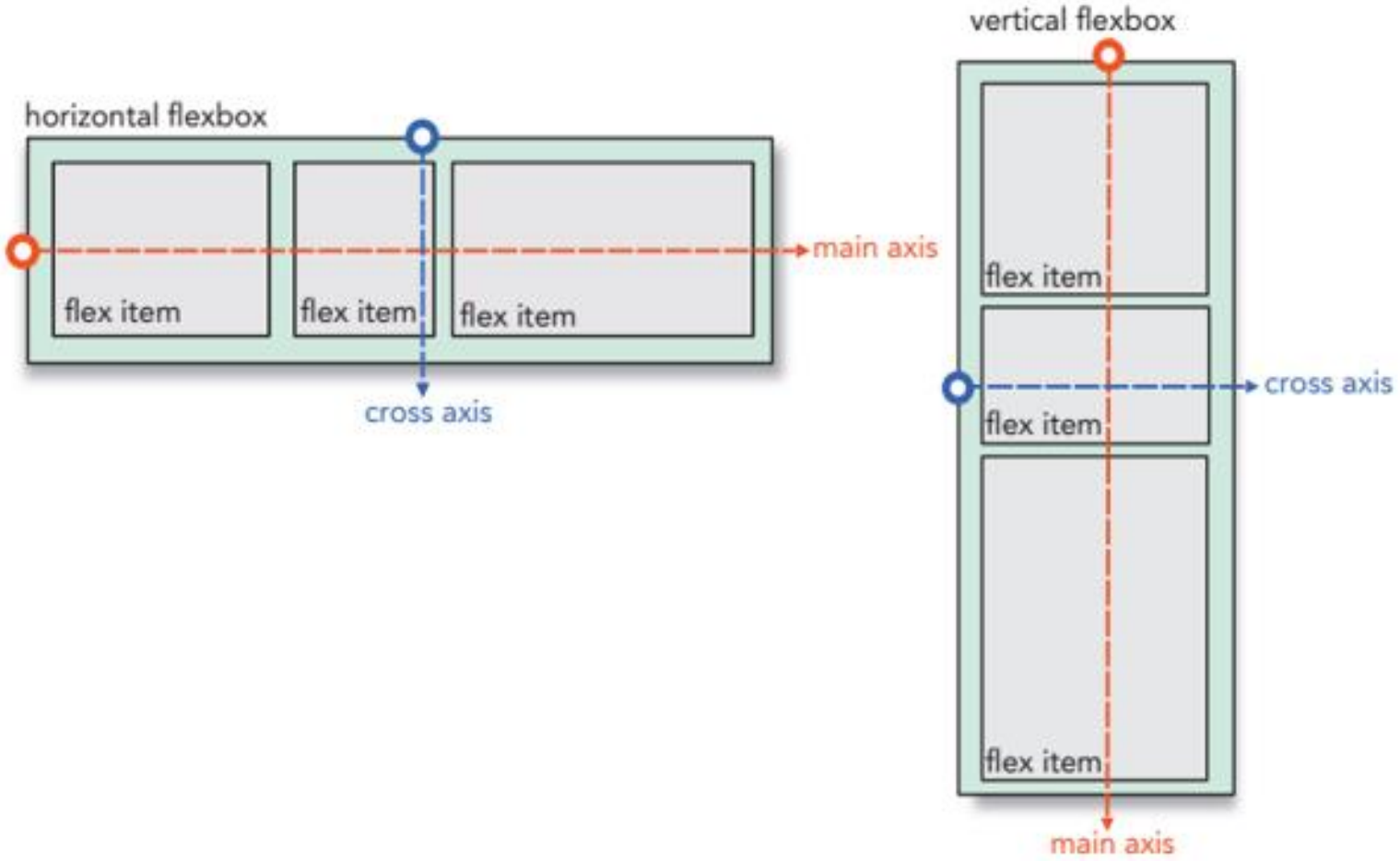
Appendix

FLEXBOX DESIGN

FlexBox Design

- A flexible box or flexbox is a box containing items whose sizes can shrink or grow to match the boundaries of the box
- Items within a flexbox are laid out along a main axis
- The **main axis** can point in either the horizontal or vertical direction
- **Cross axis** is perpendicular to the main axis and is used to define the height or width of each item

FlexBox Design (con't)



FlexBox Design (con't)

- To define an element as a flexbox, apply either of the following display styles:

```
display: flex;
```

or

```
display: inline-flex;
```

where a value of `flex` starts the flexbox on a new line and a value of `inline-flex` keeps the flexbox in-line with its surrounding content

- The complete list of browser extensions that define a flexbox is entered as:

```
display: -webkit-box;
display: -moz-box;
display: -ms-flexbox;
display: -webkit-flex:
display: flex;
```

FlexBox Design (con't)

- By default, flexbox items are arranged horizontally starting from the left and moving to the right
- The orientation of a flexbox can be changed using:

  `flex-direction: direction;`

  where `direction` is `row` (the default), `column`, `row-reverse`, or `column-reverse`
- The `row` option in a `flex-direction` lays out the flex items from left to right
- The `column` option in a `flex-direction` creates a vertical layout starting from the top and moving downward
- The `row-reverse` and `column-reverse` options in a `flex-direction` lay out the items bottom-to-top and right-to-left respectively

FlexBox Design (con't)

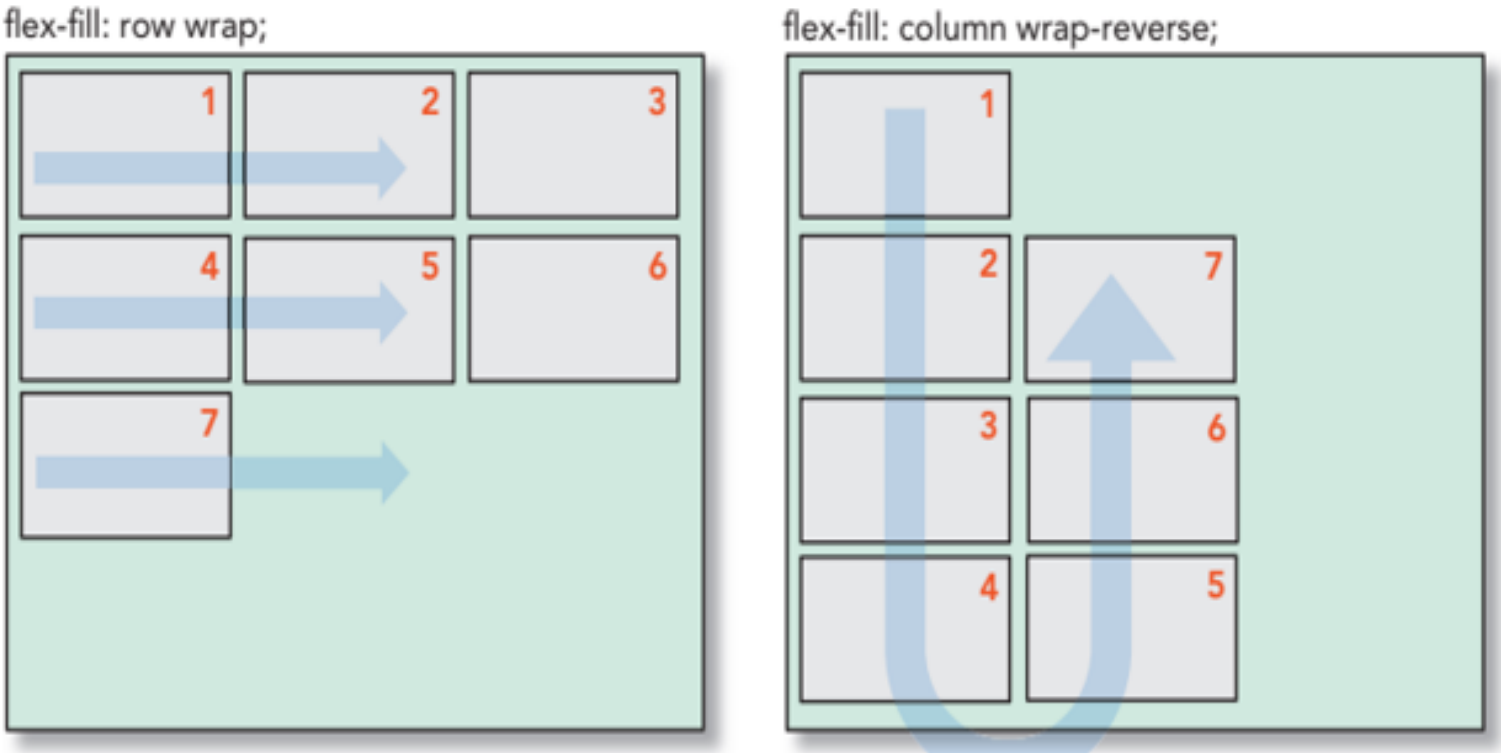
- Flex items try to fit within a single line, either horizontally or vertically
- Flex items can wrap to a new line using the following property:

  `flex-wrap: type;`

  where `type` is either:

  - `nowrap` (default)
  - `wrap` to wrap the flex items to a new line
  - `wrap-reverse` to wrap flex items to a new line starting in the opposite direction from the current line

FlexBox Design (con't)



FlexBox Design (con't)

- The flex items are determined by three properties:
  - base size
  - growth value
  - shrink value
- The basis size defines the initial size of the item before the browser attempts to fit it to the flexbox
- The basis size is set using the following:

  `flex-basis: size;`

  where `size` is one of the CSS units of measurement, which sets the initial size of the flex item based on its content or the value of its width or height property
- For example:

```
aside {flex-basis: 200px;}
```

FlexBox Design (con't)

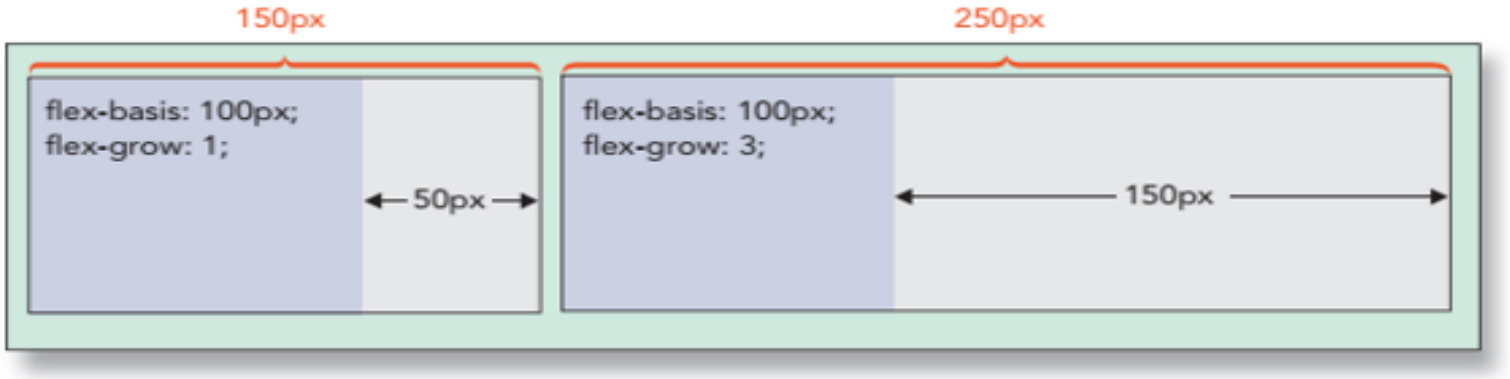
- The rate at which a flex item grows from its basis size is determined by the `flex-grow` property

  ```
  flex-grow: value;
  ```

  where `value` is a non-negative value that expresses the growth of the flex item relative to the growth of other items in the flexbox
- The default `flex-grow` value is 0, which is equivalent to the flex item remaining at its basis size

FlexBox Design (con't)

both flex items have the same basis size



second item grows at 3x the rate of the first as the flexbox expands

FlexBox Design (con't)

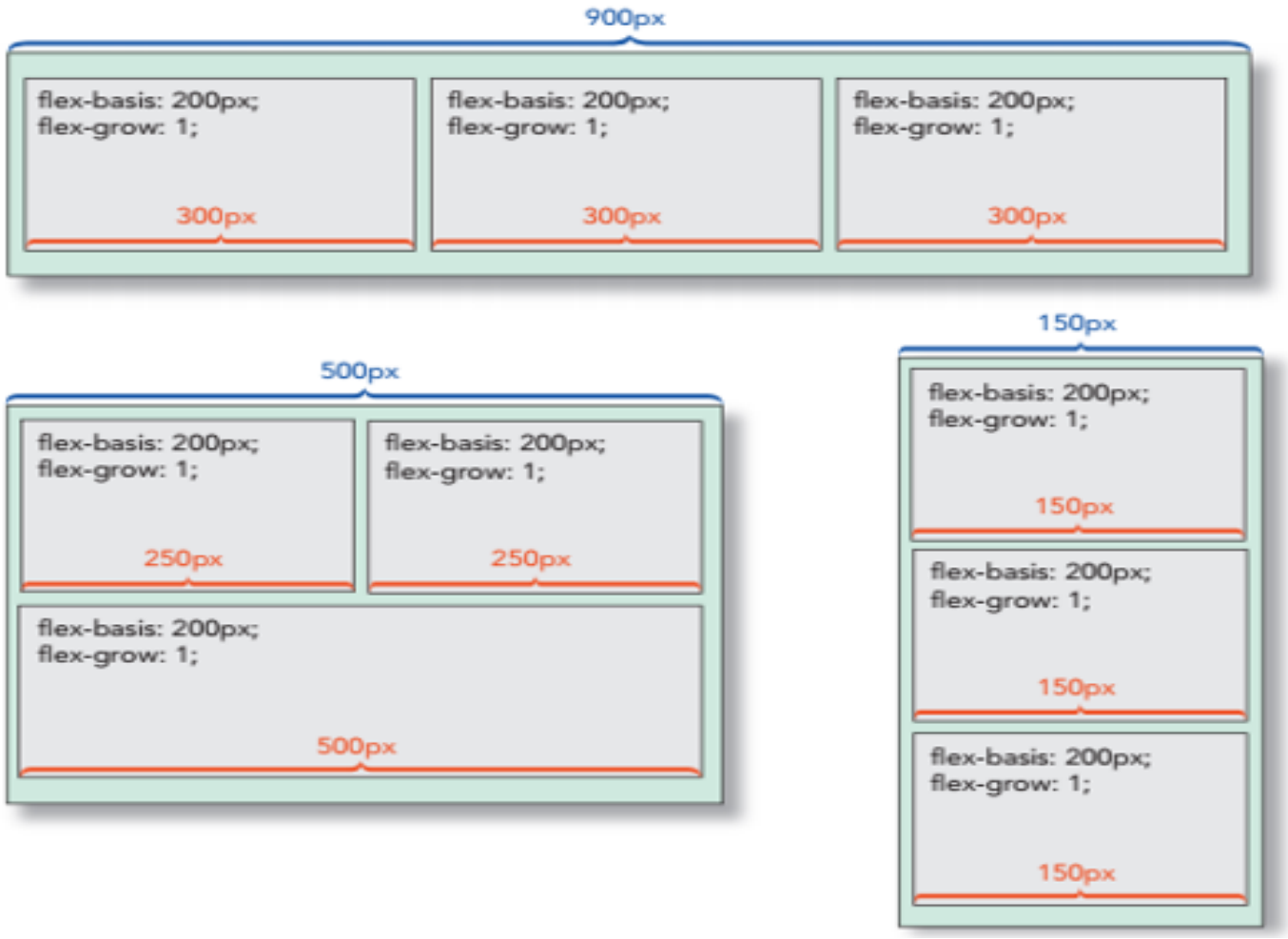
- The following style rule creates a layout for navigation list in which each list item is assigned an equal size and grows at the same rate

```
nav ul {
      display: flex;
}
nav ul li {
      flex-basis: 0px;
      flex-grow: 1;
}
```

FlexBox Design (con't)

- When the flexbox size falls below the total space allotted to its flex items:
  - Two possibilities occur depending on whether the flexbox is defined to wrap its contents to a new line
- First, if the `flexbox-wrap` property is set to `wrap`, one or more of the flex items will be shifted to a new line and expanded to fill in the available space on that line

FlexBox Design (con't)



FlexBox Design (con't)

- Second, if the flexbox does not wrap to a new line as it is resized, then the flex items will continue to shrink, still sharing the same row or column
- The rate at which flexboxes shrink below their basis size is given by the following property:

  ```
  flex-shrink: value;
  ```

  where `value` is a non-negative value that expresses the shrink rate of the flex item relative to the shrinkage of the other items in the flexbox
- The default `flex-shrink` value is 1
- If the `flex-shrink` value is set to 0, then the flex item will not shrink below its basis

FlexBox Design (con't)

- The syntax for the `flex` property is:

  ```
  flex: grow shrink basis;
  ```

  where `grow` defines the growth of the flex item, `shrink` provides its shrink rate, and `basis` sets the item's initial size

- The default flex value is:

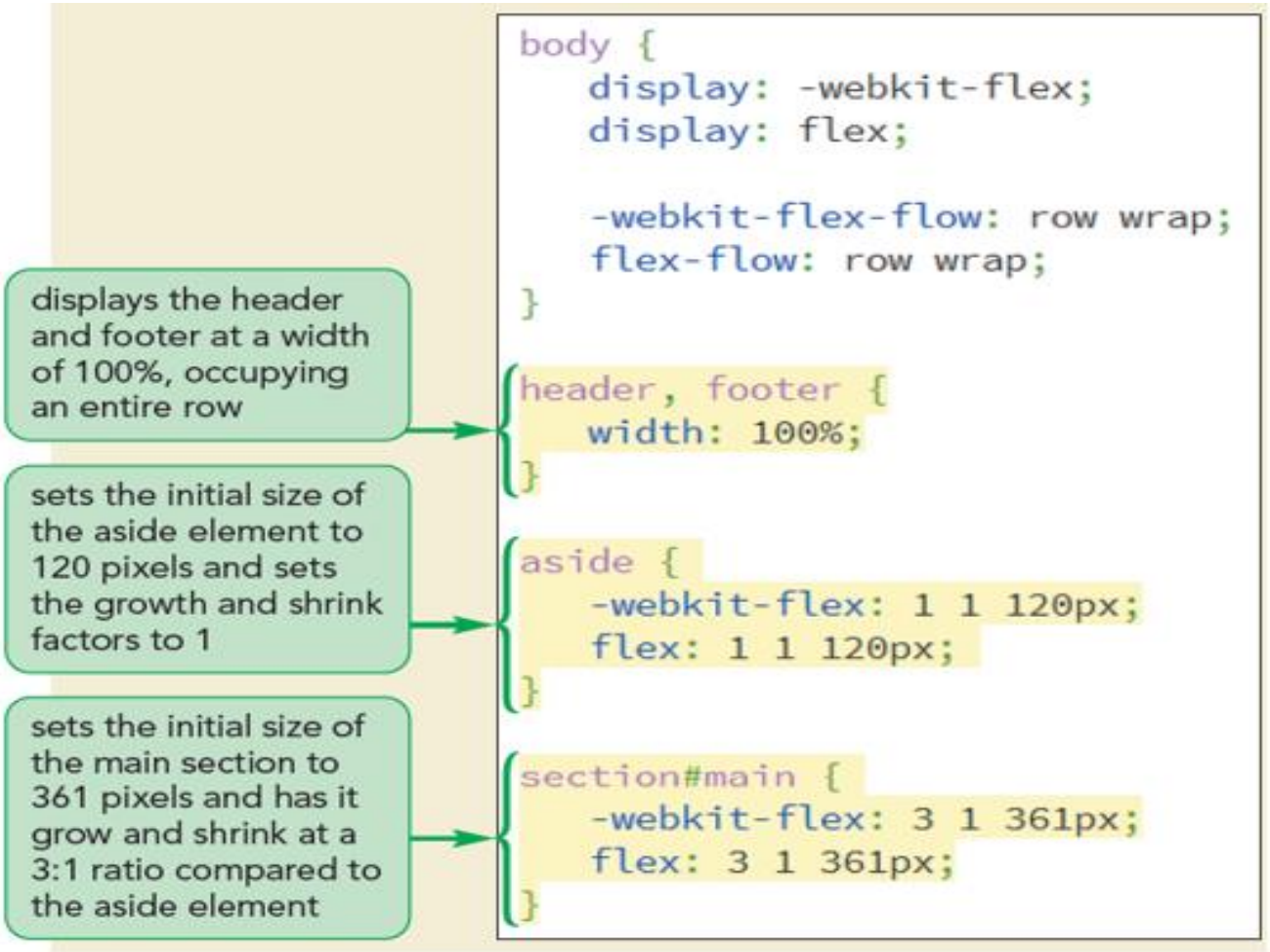
  ```
  flex: 0 1 auto;
  ```

  which automatically sets the size of the flex item to match its content or the value of its `width` and `height` property

FlexBox Design (con't)

- The `flex` property supports the following keywords:
  - `auto` – Use to automatically resize the item from its default size (equivalent to `flex: 1 1 auto;`)
  - `initial` – The default value (equivalent to `flex: 0 1 auto;`)
  - `none` – Use to create an inflexible item that will not grow or shrink (equivalent to `flex: 0 0 auto;`)
  - `inherit` – Use to inherit the flex values of its parent element

FlexBox Design (con't)



FlexBox Design (con't)

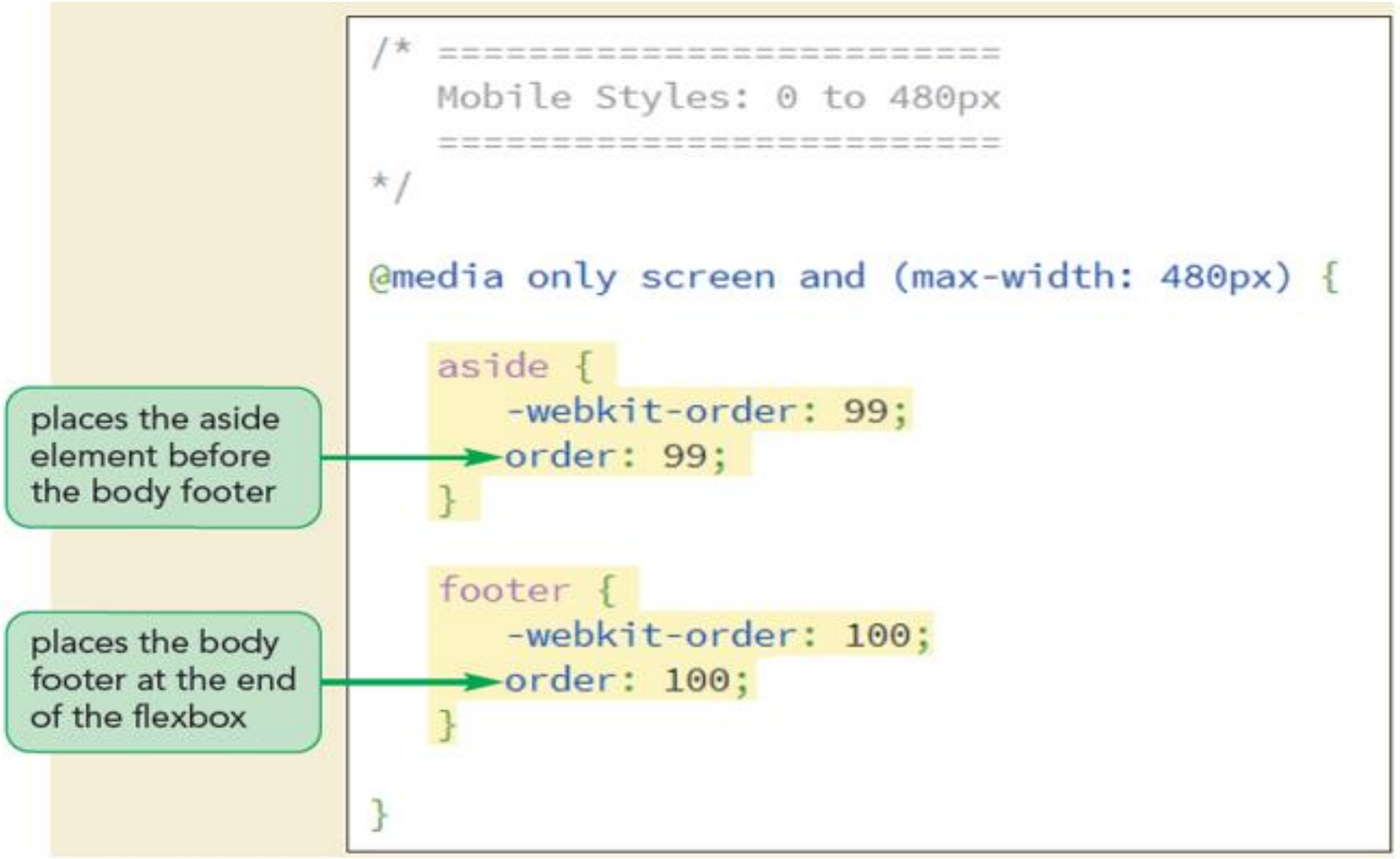
FlexBox Design (con't)

- The flexbox model allows to place the flex items in any order using the following order property:

```
order: value;
```

  where `value` is an integer where items with smaller `order` values are placed before items with larger `order` values

FlexBox Design (con't)



FlexBox Design (con't)

- By default, flex items are laid down at the start of the main axis
- To specify a different placement, apply the following `justify-content` property

  ```
  justify-content: placement;
  ```

  where `placement` is one of the following keywords:

  - `flex-start` – Items are positioned at the start of the main axis (the default)
  - `flex-end` – Items are positioned at the end of the main axis
  - `center` – Items are centered along the main axis
  - `space-between` – Items are distributed evenly with the first and last items aligned with the start and end of the main axis
  - `space-around` – Items are distributed evenly along the main axis with equal space between them and the ends of the flexbox

FlexBox Design (con't)

- The `align-content` property is similar to the `justify-content` property except that it arranges multiple lines of content along the flexbox's cross axis
- The syntax of the align-content property is:

  ```
  align-content: value;
  ```

  where `value` is one of the following keywords:

  `flex-start` - Lines are posit

  `flex-end` - Lines are positioned at the end of the cross axis

  `stretch` - Lines are stretched to fill up the cross axis (the default)

  `center` - Lines are centered along the cross axis

  `space-between` - Lines are distributed evenly with the first and last lines aligned with the start and end of the cross axis

  ioned at the start of the cross axis

  `space-around` - Lines are distributed evenly along the cross axis with equal space between them and the ends of the cross axis

FlexBox Design (con't)

- The align-items property aligns each flex item about the cross axis
- The syntax is:

  `align-items: value;`

  where `value` is one of the following keywords:

  - `flex-start` – Items are positioned at the start of the cross axis
  - `flex-end` – Items are positioned at the end of the cross axis
  - `center` – Items are centered along the cross axis
  - `stretch` – Items are stretched to fill up the cross axis (the default)
  - `baseline` – Items are positioned so that the baselines of their content align

FlexBox Design (con't)

- The `align-items` property is only impactful when there is a single line of flex items
- The `align-content` property is used to layout the flexbox content for multiple lines of flex items
- To align a single item out of a line of flex items, use the following `align-self` property:

  ```
  align-self: value;
  ```

  where `value` is one of the alignment choices supported by the `align-self` property

FlexBox Design (con't)

- **Navicon** – It is used to indicate the presence of hidden navigation menus in mobile websites
- The navicon is a symbol represented as three horizontal lines
- When a user hovers or touches the navicon, the navigation menu is revealed

```
<nav class="horizontal">
   <a id="navicon" href="#"><img src="tf_navicon.png" alt="" /></a>
   <ul>
      <li><a href="tf_home.html">Home</a></li>
      <li><a href="#">Infants</a></li>
      <li><a href="#">Toddlers</a></li>
      <li><a href="#" id="currentPage">Pre-K</a></li>
      <li><a href="#">After School</a></li>
   </ul>
</nav>
```

navicon image

Appendix

PRINTED MEDIA

Printed Media

- A **print style sheet** formats the printed version of a web document
- Browsers support their own internal style sheet to format the print versions of the web pages they encounter
  - Their default styles might not always result in the best printouts
- To apply a print style sheet, the `media` attribute is used in the `link` elements to target style sheets to either screen devices or print devices

Printed Media (con't)

Printed Media (con't)

- Every printed page in CSS is defined as a **page box**
- A page box is composed of two areas:
  - **Page area** – contains the content of the document
  - **Margin area** – contains the space between the printed content and the edges of the page

Printed Media (con't)

- Styles are applied to the page box using:

```
@page {
style rules
}
```

  where *style rules* are the styles applied to the page
- The styles are limited to defining the page size and the page margin

Printed Media (con't)

- The following `size` property allows web authors to define the dimensions of a printed page:

  `size:` *`width height`*`;`

  where *`width`* and *`height`* are the width and height of the page
- The keyword `auto` lets browsers determine the page dimensions
- The keyword `inherit` inherits the page size from the parent element

Printed Media (con't)

- Different styles can be defined for different pages by adding the following:

```
@page:pseudo-class {
        style rules
}
```

where *pseudo-class* is `first` for the first page of the printout, `left` for the pages that appear on the left in the double-sided printouts, or `right` for pages that appear on the right in double-sided printouts

Printed Media (con't)

- To define styles for pages other than the first, left, or right, create a page name as follows:

```
@page name {
        style rules
}
```

where *name* is the label given to the page

- To assign a page name to an element, use

```
selector {
        page: name;
}
```

where *selector* identifies the element that will be displayed on its own page, and *name* is the name of a previously defined page style

Printed Media (con't)

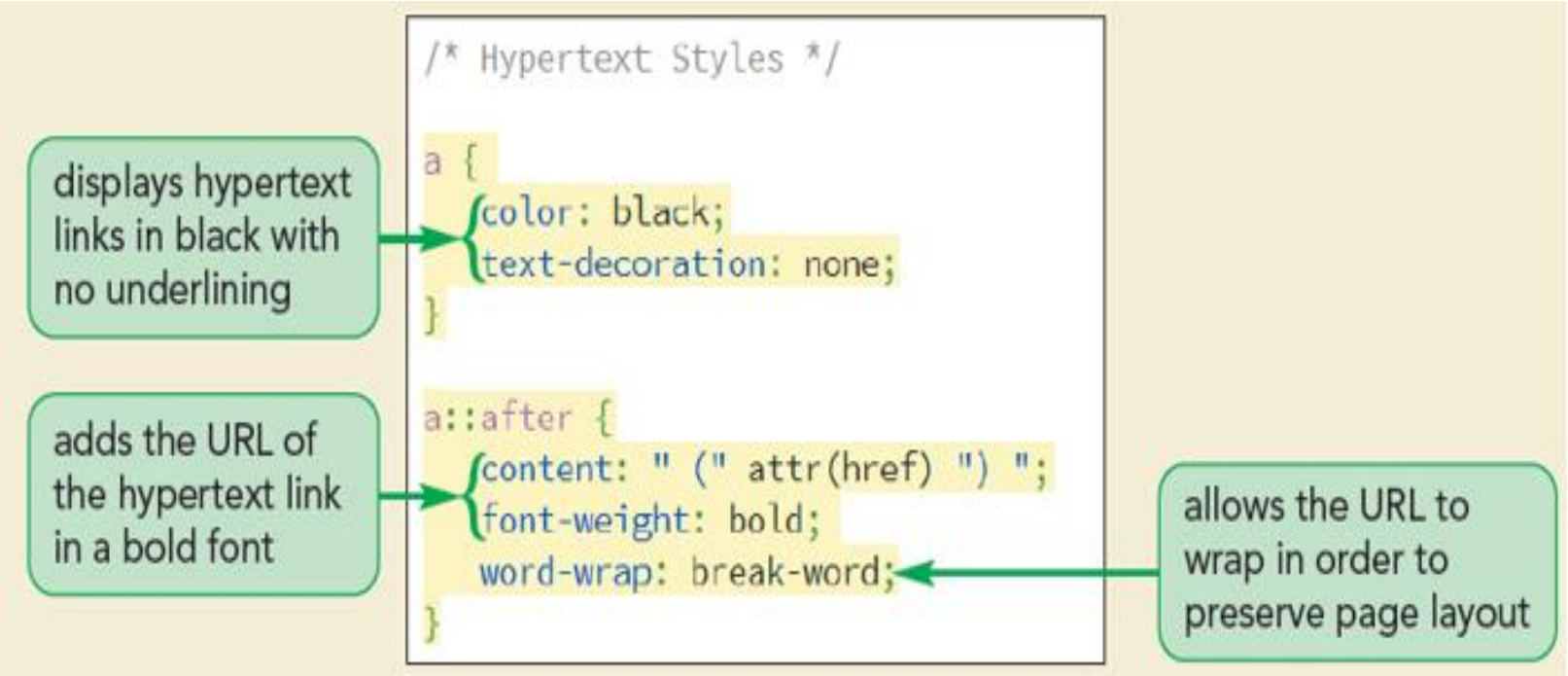
- To append the text of a link's URL to the linked text, apply the following style rule:

```
a::after {
        content: " (" attr(href) ") ";
}
```

This style rule uses the `after` pseudo-element along with the `content` property and the `attr()` function to retrieve the text of the `href` attribute and add it to the contents of the `a` element

The `word-wrap` property is used to break long text strings at arbitrary points if it extends beyond the boundaries of its container

Printed Media (con't)



Printed Media (con't)

- Page breaks can be inserted either directly before or after an element, using the following properties:

```
page-break-before: type;
page-break-after: type;
```

where *type* has the following possible values:

- `always` – Use to always place a page break before or after the element
- `avoid` – Use to never place a page break
- `left` – Use to place a page break where the next page will be a left page
- `right` – Use to place a page break where the next page will be a right page
- `auto` – Use to allow the printer to determine whether or not to insert a page break
- `inherit` – Use to insert the page break style from the parent element

Printed Media (con't)

- Page breaks can be prevented by using the keyword `avoid` in the `page-break-after` or `page-break-before` properties
- For example, the following style rule prevents page breaks from being added after any heading

```
h1, h2, h3, h4, h5, h6 {
    page-break-after: avoid;
}
```

Printed Media (con't)

- Page breaks within block elements, such as paragraphs, often leave behind widows and orphans
- A widow is a fragment of text left dangling at the top of a page
- An orphan is a text fragment left at the bottom of a page
- To control the size of widows and orphans, CSS supports the following properties:

```
widows: value;
orphans: value;
```

where *value* is the number of lines that must appear within the element before a page break can be inserted by printer

Appendix

MOBILE TABLES

Tables and Mobile Devices

- Tables do not scale well to mobile devices
- Problems faced by users to view a table in a mobile device
  - Table is too small to read
  - Table does not fit the visual viewport
  - Table columns are too narrow to read the cell content

complete table is too small to read

table is easier to read but does not fit within the viewport

table fits within the viewport but columns are too narrow to read comfortably

Tables and Mobile Devices (con't)

- A new layout of table data for mobile screens is required
- Several table columns are reduced to two:
  - One column containing all data labels
  - Second column containing data associated with each label
- To create a responsive web table, add the text of data labels as attributes of all `td` elements in the table body
- Store data labels using a **data attribute**
- General format of a data attribute is

  `data-text="value"`

  where `text` is the name of the data attribute and `value` is its value

Tables and Mobile Devices (con't)

- Data attributes use names specific to the function it is used for
- For example, the following code uses a data attribute named `data-label` to store the text of the labels associated with the data cell:

```
<td data-label="Date">April 2,   2021</td>
```

- The result is a list of data cells that are aligned as block elements
- Within each block element, the data label is followed by the data cell content
- The goal is to transform table with multiple columns into two-column layout

Tables and Mobile Devices (con't)

Tables and Mobile Devices (con't)

- Column layout enables display of content side-by-side in a page
- Layouts that use float elements or flexboxes differ from column layout
  - Single element can flow from one column to the next
  - Flow of content adjusts to match the page width
- Size of a column is set using the `column-count` property

  ```
  column-count: value;
  ```

  where `value` is the number of columns in the layout
- Browser extensions are included to ensure cross-browser compatibility

Tables and Mobile Devices (con't)

- Columns are laid out evenly across the width of the parent element by default
- To set the column width, use the `column-width` property

  `column-width: size;`

  where `size` is the minimum width of the column
- Column width acts like the basis value for items in a flexbox

Tables and Mobile Devices (con't)

- The `column-width` and `column-count` properties are combined to form shorthand columns property

```
columns: width count;
```

- The default gap between columns is 1em
- To set a different gap size, use the `column-gap` property

```
column-gap: size;
```

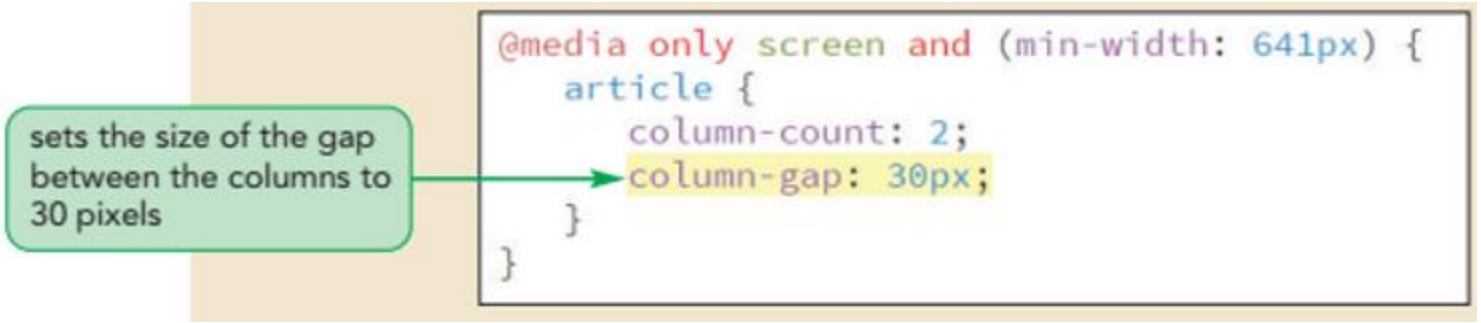
where `size` is the width of the gap

Tables and Mobile Devices (con't)

- Another way to separate columns is with a graphic dividing line created using the `column-rule` property

  `column-rule: border;`

  where `border` defines the style of dividing line
- The `column-rule` property can be broken into individual properties like `column-rule-width`, `column-rule-style`, and `column-rule-color`



Tables and Mobile Devices (con't)

- The size of column orphans is controlled using the `orphans` property

  ```
  orphans: value;
  ```

  where `value` is the minimum number of lines stranded before a column break

- The size of column widows is controlled using the `widows` property

  ```
  widows: value;
  ```

  where `value` is the minimum number of lines placed after a column break

Tables and Mobile Devices (con't)

- Other properties to define column breaks

  `break-before: type;`

  `break-after: type;`

  where `type` is one of the following:

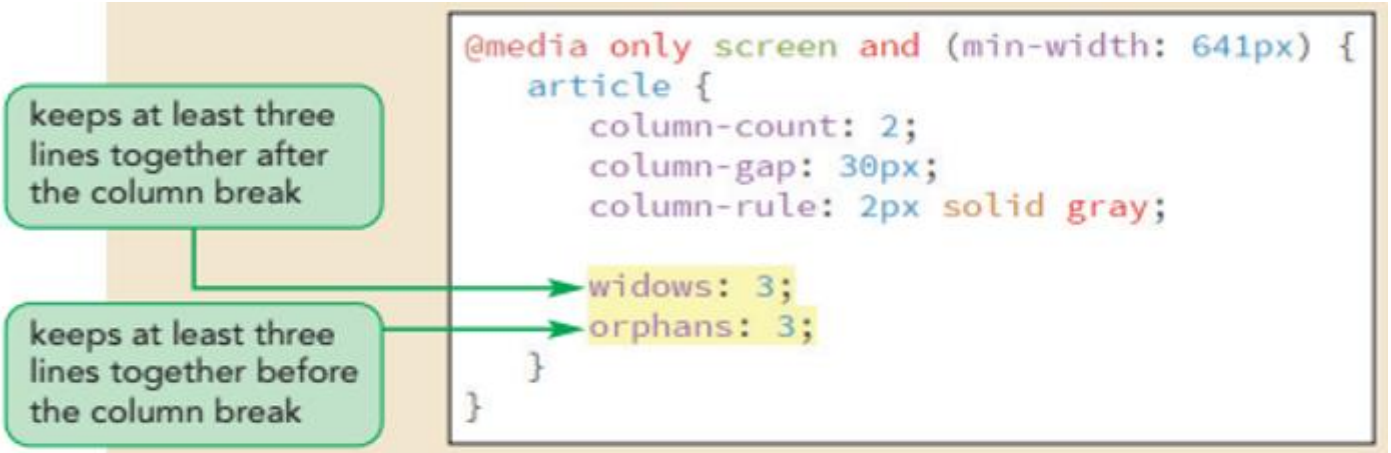
  - `auto` (browser automatically sets column break)
  - `always` (to always place a column break)
  - `avoid` (to avoid placing a column break)

Tables and Mobile Devices (con't)

- To control placement of column breaks within an element, use the property

  `break-inside: type;`

  where *type* is either `auto` or `avoid`

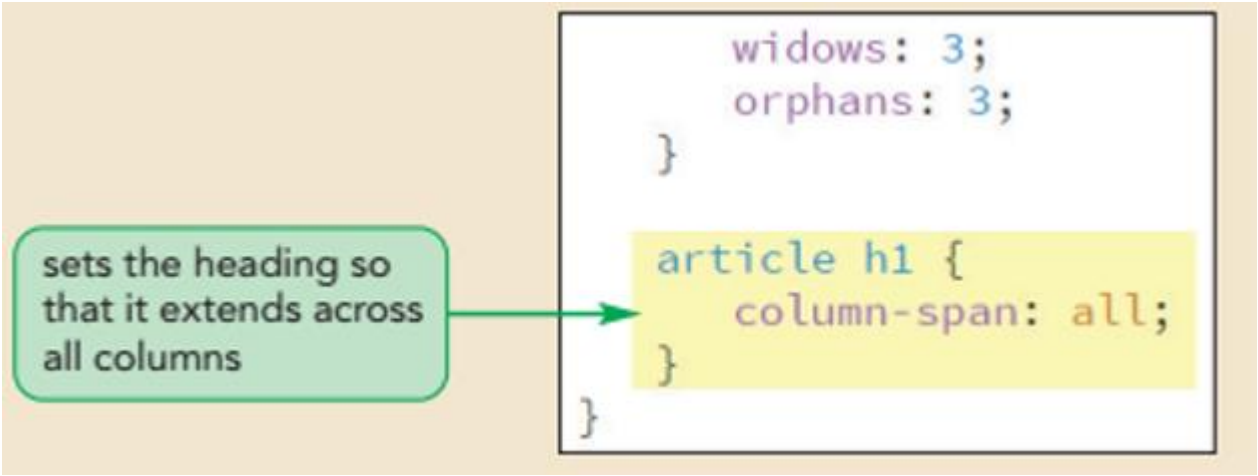


Tables and Mobile Devices (con't)

- To span cell columns, use the `column-span` property

  `column-span: span;`

  where `span` is either `none` to prevent spanning or `all` to enable the content to span across all the columns



Spanning Cell Columns (continued)

Figure 6–46 Creating a column-spanning heading